

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»

# ТЕХНІЧНІ ЗАСОБИ ІНТЕРНЕТУ РЕЧЕЙ

## ЛАБОРАТОРНИЙ ПРАКТИКУМ

*Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського  
як навчальний посібник для студентів,  
які навчаються за спеціальністю 171 «Електроніка»,  
освітня професійна програма «Електронні системи мультимедіа та засоби  
Інтернету речей»*

Київ  
КПІ ім. Ігоря Сікорського  
2020

Технічні засоби Інтернету речей [Електронний ресурс] : навч. посіб. для студ. спеціальності 171 «Електроніка», спеціалізації «Електронні системи мультимедіа та засоби Інтернету речей» / КПІ ім. Ігоря Сікорського; уклад.: Ю.О.Оникієнко, О.О. Титаренко. – Електронні текстові данні (1 файл: 10,8 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2020. –124 с.

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 18.06.2020 р.)  
за поданням Вченої ради факультету електроніки (протокол № 5/2020 від 25.05.2020 р.)*

Електронне мережне навчальне видання

# **ТЕХНІЧНІ ЗАСОБИ ІНТЕРНЕТУ РЕЧЕЙ**

## **ЛАБОРАТОРНИЙ ПРАКТИКУМ**

Укладачі: *Оникієнко Юрій Олексійович, канд. тех. наук, доц.  
Титаренко Ольга Олександрівна*

Відповідальний: *Лазебний Володимир Семенович, доцент кафедри*  
редактор *Акустичних та Мультимедійних Електронних Систем, к.т.н.,  
доцент*

Рецензент: *Клен Катерина Сергіївна, канд. тех. наук, доцент*

В роботі наведено загальні відомості про Інтернет речей. Описано особливості реалізації архітектури та концепції Інтернету речей. Детально розглянуті основні платформи Інтернету речей, що надають можливість поєднувати та підключати пристрої до глобальної мережі Інтернет для обміну та моніторингу даних. Описано основні протоколи, що використовує Інтернет речей. В роботі наведено дев'ять лабораторних робіт різного ступеня складності, які базуються на програмно-апаратному комплексі Arduino. В результаті виконання курсу лабораторних робіт студенти зможуть писати та налагоджувати програми у середовищі розробки Arduino для реалізації рішень і додатків з використанням концепції Інтернету речей.

© Ю.О.Оникієнко, О.О. Титаренко, 2020

© КПІ ім. Ігоря Сікорського, 2020

## ЗМІСТ

РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНТЕРНЕТ РЕЧЕЙ.....	5
1.1 Концепція IoT.....	5
1.2 Архітектура IoT.....	8
1.3 Платформи IoT.....	10
РОЗДІЛ 2 ПРОТОКОЛИ ІОТ.....	18
2.1 Протокол MQTT .....	18
2.2 Протокол BLE .....	28
2.3 Протокол LoRaWAN .....	36
2.4 Протокол Ethernet.....	41
РОЗДІЛ 3 ЛАБОРАТОРНИЙ ПРАКТИКУМ.....	44
3.1 Лабораторна робота №1. Вивчення роботи датчика тиску BMP180.....	44
3.2 Лабораторна робота №2. Взаємодія контролера Arduino за програмою Node-RED .....	49
3.3 Лабораторна робота №3. Підключення Wi-Fi модуля ESP8266 до сервера Thingier.io.....	56
3.4 Лабораторна робота №4. Підключення модуля ESP8266 до платформи Ubidots через HTTP.....	64
3.5 Лабораторна робота №5. Реалізація протоколу MQTT за допомогою Arduino .....	71
3.6 Лабораторна робота №6. Bluetooth модуль ESP32.....	81
3.7 Лабораторна робота №7. Створення сервера на базі Arduino Uno за допомогою W5100 Ethernet Shield.....	90
3.8 Лабораторна робота №8. Вивчення роботи модуля LoRa.....	98

3.9	Лабораторна робота №9. Підключення модуля ESP8266 NodeMcu V3 до IoT платформи ThingSpeak з використанням AT команд.....	114
	ПЕРЕЛІК ПОСИЛАНЬ .....	123

## РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНТЕРНЕТ РЕЧЕЙ

### 1.1 Концепція IoT

Термін "Інтернет речей" (Internet of Things, IoT) був запропонований в 1999 році Кевіном Ештоном, одним з трьох засновників Центру автоматичної ідентифікації Массачусетського університету (Auto-ID Center).

Інтернет речей – це глобальна мережа підключених до Інтернету фізичних пристроїв ("речей"), оснащених сенсорами, датчиками і пристроями передавання інформації.

Саме поняття "Інтернет речей" являє собою цілу концепцію комунікаційної мережі об'єктів, що мають технології для взаємодії між собою та з навколишнім середовищем. Крім того, вони здатні виконувати певні дії без втручання людини. Кожному фізичному пристрою мережі (або групі пристроїв) надають IP-адресу, постійну або тимчасову (динамічну).

В більшості джерел зазначено, що попередником IoT є концепція міжмашинної комунікації Machine-to-Machine (M2M), яку образно називають "Інтернетом машин". Інтернет речей містить M2M як складову і є подальшим технологічним розвитком раніше закладених ідей і їх маркетинговим переосмисленням.

Парадигму M2M можна уявити як "сплав" можливостей обумовлених сучасними технологіями:

- безпроводові комунікації;
- значна обчислювальна потужність у вузлах, пристроях M2M;
- мережні («хмарні») сервіси і технології.

Кожен з трьох компонентів робить свій внесок у формування цінності для споживача і постачальника рішень:

*Безпроводові комунікації* – суттєве скорочення термінів проектування і монтажу, можливість гнучкого переналаштування, в тому числі і на перспективні завдання.

*Обчислювальна потужність у вузлах, пристроях M2M* – дозволяє на рівні датчиків і сенсорів використовувати багато недорогих пристроїв. Від оброблення "в центрі" можна перейти до оброблення "на місці".

*Хмарні сервіси і технології* гарантують максимальну гнучкість і знижують не тільки рівень користувацьких витрат, а й "порог" входження в нові рішення.

Додаткові блоки, які лягли в фундамент Інтернету речей:

- низька вартість користування;
- мінімальне енергоспоживання;
- комунікації і базова функціональність, доступна "з коробки";
- глобальна доступність пристроїв і сервісів.

Слід розрізняти поняття "Інтернет речей" і "Інтернет-річ". Під Інтернет-річчю розуміють будь-який пристрій, який:

- має доступ до мережі Інтернет з метою передавання або запиту будь-яких даних;
- має конкретну адресу в глобальній мережі або ідентифікатор, за яким можна здійснити зворотний зв'язок з річчю;
- має інтерфейс для взаємодії з користувачем.

Інтернет-речі мають єдиний протокол взаємодії, згідно з яким будь-який вузол мережі рівноправний в наданні своїх сервісів.

На шляху переходу до втілення ідеї Інтернету речей стояла проблема, пов'язана з протоколом IPv4, ресурс вільних мережевих адрес якого вже практично вичерпав себе. Однак підготовка до повсюдного впровадження версії протоколу IPv6 дозволяє вирішити цю проблему і наближає ідею Інтернету речей до реальності, надаючи можливість використання кожним жителем Землі більше 300 млн. IP-адрес.

Кожен вузол мережі Інтернет-речей має свій сервіс, надаючи якусь послугу поставки даних. У той же час вузол такої мережі може приймати команди від будь-якого іншого вузла. Це означає, що всі Інтернет-речі можуть

взаємодіяти одна з одною і вирішувати спільні обчислювальні завдання. Інтернет-речі можуть утворювати локальні мережі, об'єднанні певною зоною обслуговування або функцією.

На рис. 1 наведено точку зору компанії Beecham Research. Класи додатків групуються по секторам вертикальних ринків, а в кожному з них перераховується можливе місце застосування і види пристроїв.

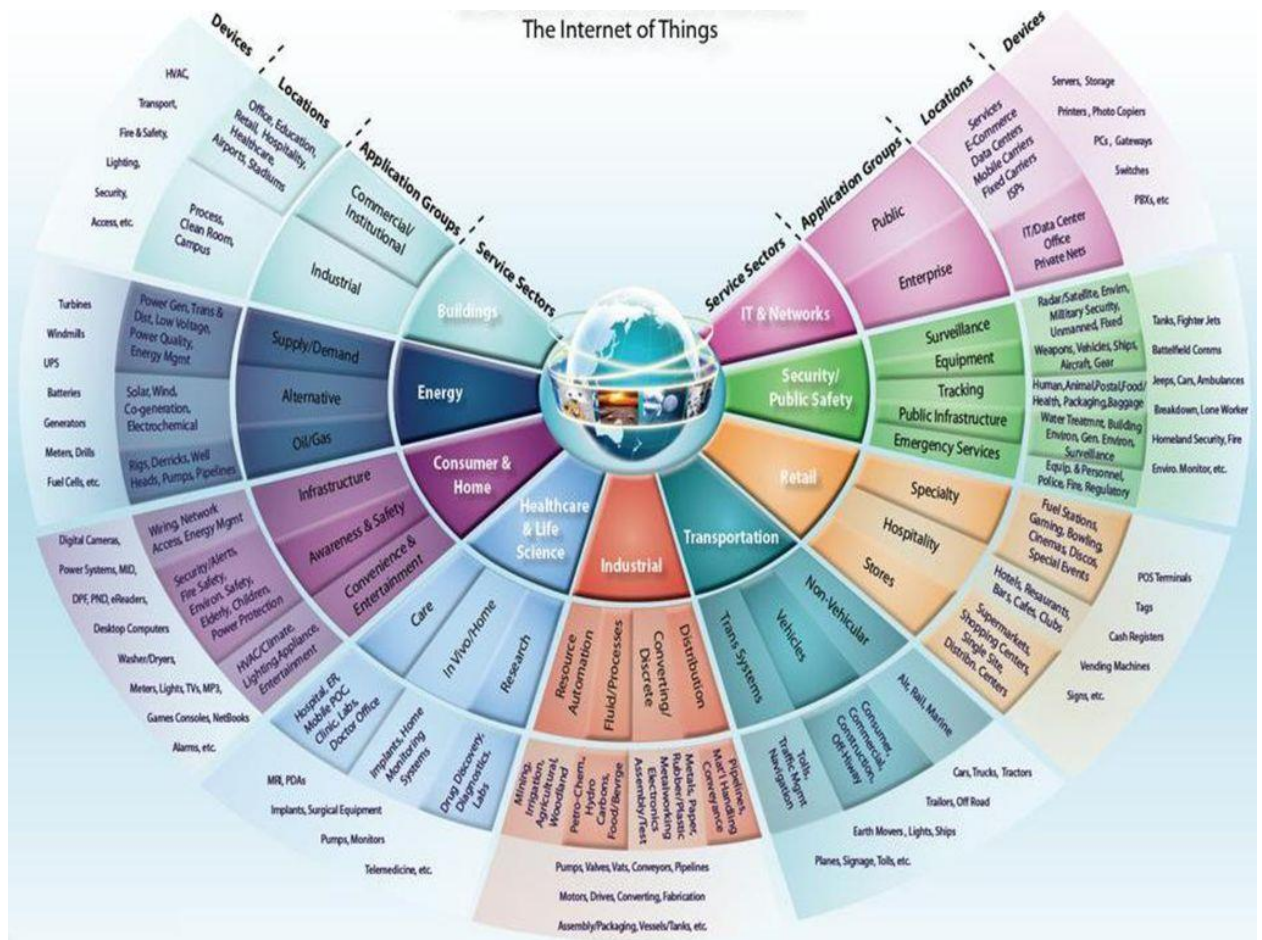


Рисунок 1.1 – Світ M2M і IoT за версією Beecham Research

Серед технологій майбутнього, які вже змінюють або змінять наше життя в найближчі роки, найбільш часто називають такі:

- штучний інтелект і розвиток робототехніки (AI);
- хмарні технології та сервіси (Cloud);
- великі дані (Big Data);

- Інтернет 3.0 (Web 3.0);
- адитивні технології (3D printing або additive manufacturing, AM).

Для перших чотирьох технологій зі списку Інтернет речей є одночасно і рушійною силою і «бенефіціаром» їх розвитку. Іншими словами, IoT в його сучасному розумінні не існує без використання хмарних сервісів, великих даних і можливостей по їх інтелектуальній обробці. Нові споживчі якості і можливості з'являються як складний сплав сучасних трендів, причому вірні пропорції для успіху знайти не просто [1].

## 1.2 Архітектура Інтернету речей

Класична архітектура інтернету речей складається з IoT-пристроїв, щлюзів, серверу та клієнтської частини. У загальному вигляді схема такого сервісу представлена на рис. 1.2.

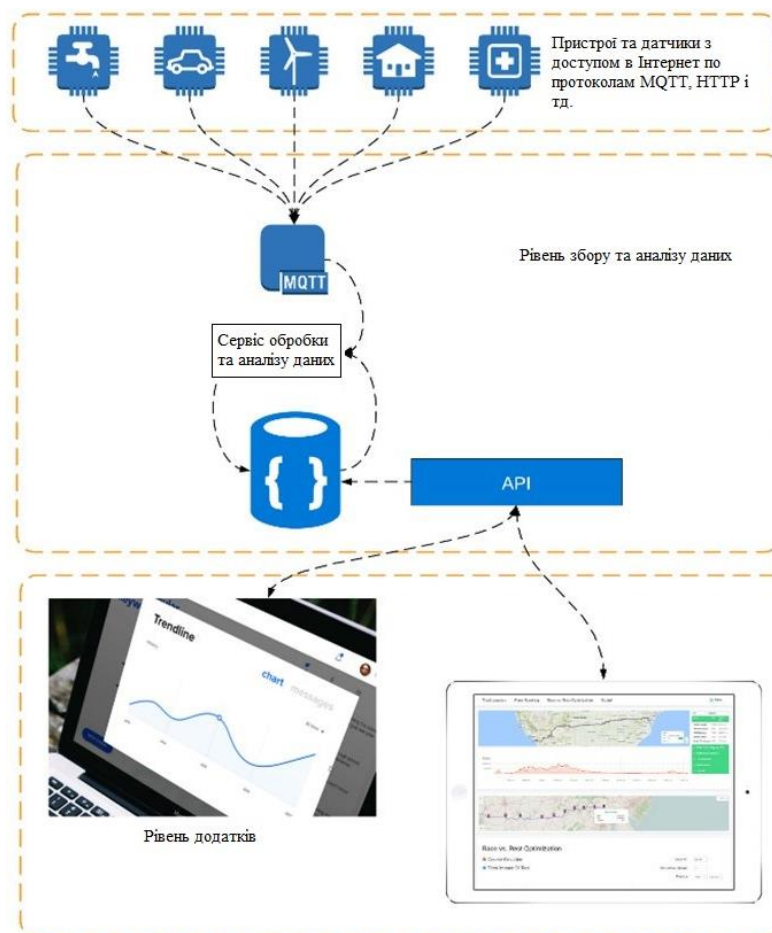


Рисунок 1.2 – Архітектура IoT



IoT-пристрої збирають свідчення з датчиків і виконують фізичні дії. Сенсори забезпечують поєднання фізичного і віртуального (цифрового) світів, виконуючи збір та обробку інформації в реальному часу. Мініатюризація, що призвела до зменшення фізичних розмірів апаратних сенсорів, дозволила інтегрувати їх безпосередньо в об'єкти фізичного світу. Існують різні типи сенсорів для відповідних цілей, наприклад, для вимірювання температури, тиску, швидкості руху, місця розташування та ін. Сенсори можуть мати невелику пам'ять, даючи можливість записувати кілька результатів вимірювань. Сенсори класифікуються відповідно до їх призначення, наприклад, сенсори навколишнього середовища, сенсори для тіла, сенсори для побутової техніки, сенсори для транспортних засобів і т.д. Більшість сенсорів вимагає з'єднання з агрегатором сенсорів (шлюзом), які можуть бути реалізовані з використанням локальних обчислювальних мереж (LAN, Local Area Network), таких як Ethernet і Wi-Fi або персональних мереж (PAN, Personal Area Network), таких як ZigBee, Bluetooth і ультраширокополосного бездротового зв'язку на малих відстанях (UWB, Ultra-Wide Band). Для сенсорів, які не вимагають підключення до агрегатору, їх зв'язок з серверами або додатками може відбуватися з використанням глобальних безпроводових мереж WAN, таких як GSM, GPRS і LTE. Сенсори, які характеризуються низьким енергоспоживанням і низькою швидкістю передачі даних, утворюють широко відомі безпроводові сенсорні мережі (WSN, Wireless Sensor Network). WSN набирають все більшої популярності, оскільки вони можуть містити набагато більше сенсорів з підтримкою роботи від батарей і охоплюють великі площі.

Шлюзи отримують інформацію від пристроїв і передають їм команди для виконання дій. Як правило, представлені апаратним маршрутизатором або програмним забезпеченням та використовують різні протоколи.

Даний рівень складається з конвергентної мережевої інфраструктури, яка створюється шляхом інтеграції різномірних мереж в єдину мережеву платформу. Конвергентний абстрактний мережевий рівень в IoT дозволяє

через відповідні шлюзи декільком користувачам використовувати ресурси в одній мережі незалежно і спільно без шкоди для конфіденційності, безпеки і продуктивності [2].

Сервер зберігає, обробляє та аналізує показники датчиків. Може бути реалізований на базі віртуального сервера, реальної машини або через хмару.

Клієнтська частина, реалізується через мобільний або веб-додаток. Забезпечує доступ до даних пристроїв і наочне уявлення результатів аналізу.

### 1.3 Платформи IoT

Стрімке зростання кількості пристроїв спонукає до пошуку рішень, що дозволять підключити ці ж пристрої до мережі, збирати, зберігати та аналізувати їх дані.

**Amazon Web Services (AWS)** пропонує широкі можливості, що охоплюють безліч компонентів – від периферійних пристроїв до хмарних систем та надає можливість створювати IoT-рішення практично для будь-яких пристроїв і прикладів використання.

AWS IoT (рис. 1.3) працює на основі хмари AWS, яку використовують провідні компанії по всьому світу, та легко масштабується в міру зростання кількості пристроїв і потреб бізнесу.

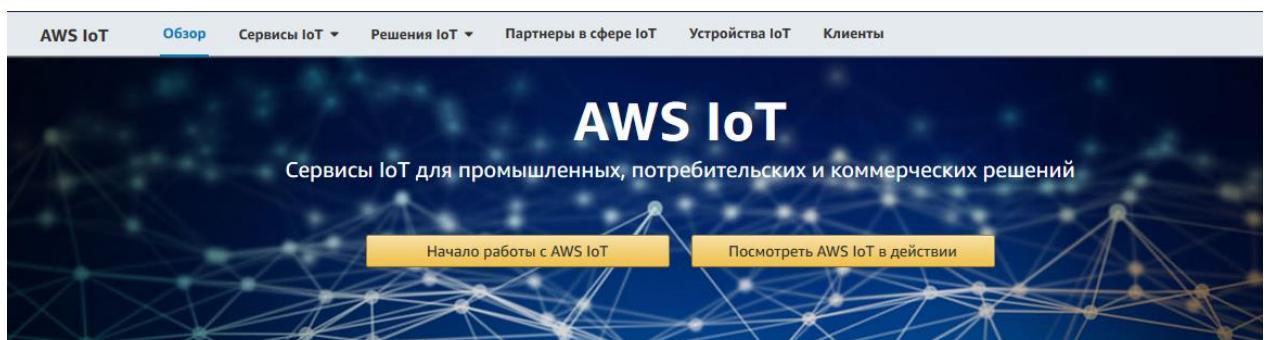


Рисунок 1.3 – Платформа Amazon Web Services IoT

Завдяки AWS IoT можна з легкістю поєднувати, управляти пристроями і збирати дані. AWS зможе підтримувати будь-які хмарні проекти практично з 100% вірогідністю. Хмарні сервіси, що надаються Amazon, включають в себе пакет IoT, який підтримує всі аспекти додатків Інтернету.

AWS IoT Core – основа, на якій може бути побудовано будь-який додаток IoT. Через AWS IoT Core пристрої можуть підключатися до Інтернету, один до одного і обмінюватися даними. Мільярди повідомлень можуть бути відправлені між пристроями і хмарним сховищем через безпечне з'єднання. Платформа підтримує різні протоколи зв'язку, в тому числі і призначені для користувача, що дозволяє здійснювати зв'язок між пристроями різних виробників.

AWS IoT Device Management дозволяє легко додавати і організовувати пристрої. Сервіс забезпечує безпечну і масштабовану продуктивність з можливістю моніторингу, усунення несправностей та оновлення функціональності пристрою.

AWS IoT Analytics надає сервіс для автоматичної аналітики великих обсягів різних даних IoT, включаючи неструктуровані дані з різних типів пристроїв. Дані, зібрані і оброблені службою, готові для використання в машинному навчанні.

AWS IoT Device Defender підтримує налаштування механізмів безпеки для систем IoT. AWS IoT Device Defender дозволяє налаштовувати і управляти політикою безпеки, контролюючи аутентифікацію і авторизацію пристрою, а також забезпечуючи механізми шифрування [3].

**Google Cloud Platform** – глобальний хмарний постачальник, що підтримує рішення IoT. Його пакет Google Cloud IoT дозволяє створювати і управляти системами IoT будь-якого розміру і складності (рис 1.4).

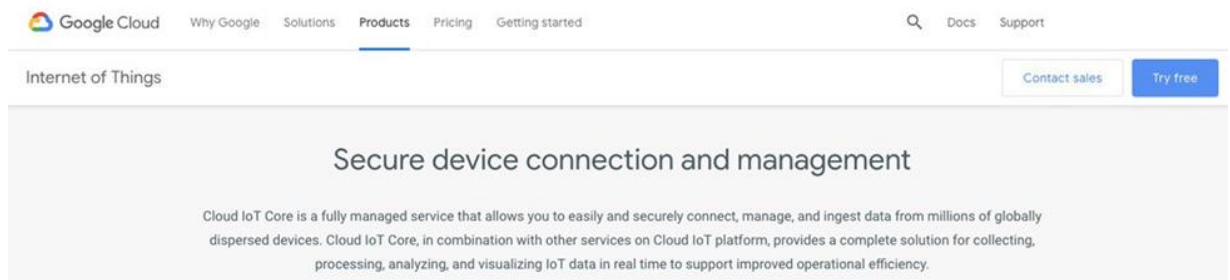


Рисунок 1.4 – Платформа Google Cloud IoT

Рішення Google Cloud IoT включає в себе ряд служб, за допомогою яких можна створювати мережі IoT:

- Cloud IoT Core – повністю керований сервіс для простого і безпечного підключення, а також управління і прийому даних з різних пристроїв;
- Cloud Pub / Sub – сервіс, який обробляє дані про події та надає аналітику потоків в реальному часі;
- Cloud Machine Learning Engine, що дозволяє створювати моделі ML і використовувати дані, отримані за допомогою пристроїв IoT.

Рішення IoT, розроблене Google, включає в себе ряд інших послуг, які можуть бути корисні при побудові комплексних підключених мереж.

**Microsoft Azure** – платформа, що пропонує, як попередньо сконфігуровані рішення, так і можливість налаштовувати їх та створювати нові, відповідно до вимог проекту (рис. 1.5).

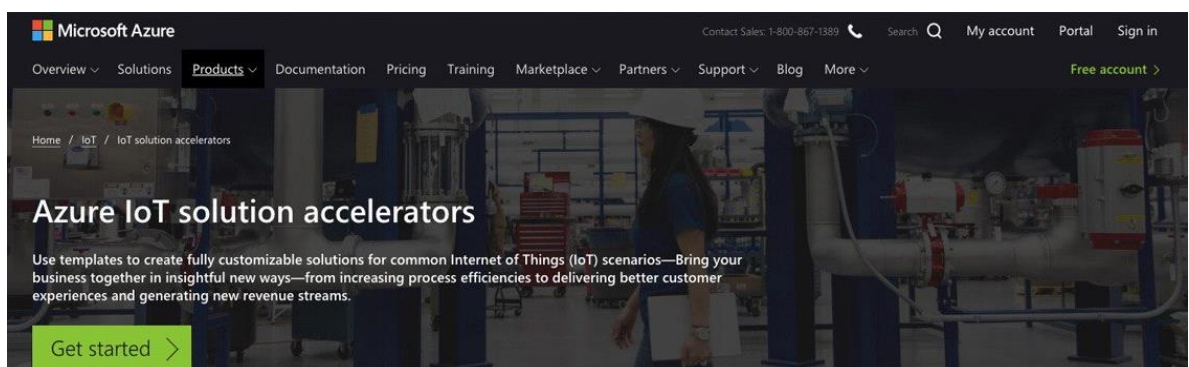


Рисунок 1.5 – Платформа Microsoft Azure IoT Suite

Microsoft Azure IoT Suite надає механізми безпеки, масштабованість і просту інтеграцію з будь-якими існуючими або майбутніми системами. Платформа дозволяє підключати сотні пристроїв різних виробників, збирати аналітичні дані і використовувати дані IoT для цілей машинного навчання.

Хмарна платформа **SAP** для Інтернету речей має все необхідне для створення і управління IoT-додатками.

Платформа SAP – це зручне середовище для віддаленого управління і моніторингу всіх підключених пристроїв, що належать системі IoT (рис. 1.6).

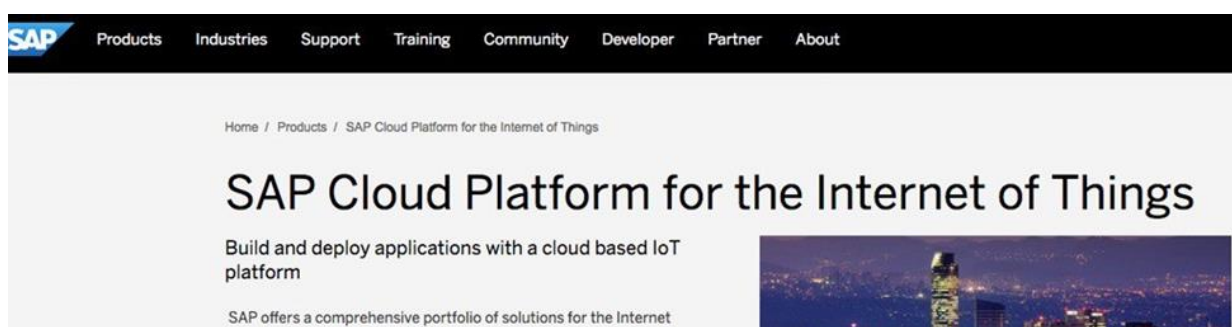


Рисунок 1.6 – Платформа Microsoft Azure IoT Suite

Дистанційні пристрої можуть бути підключені або безпосередньо, або через хмарний сервіс. Потужні аналітичні можливості дозволяють обробляти, систематизувати і вивчати дані, отримані від датчиків, лічильників та інших пристроїв IoT.

Звичайно, відповідно до останніх технологічних досягнень SAP надає можливість використовувати дані IoT для створення додатків штучного інтелекту і машинного навчання.

Платформа **Salesforce** (рис. 1.7) зосереджує свої зусилля з розробки IoT на створенні інтегрованої системи, що з'єднує пристрої IoT зі своїми клієнтами прямо в структурі Salesforce.

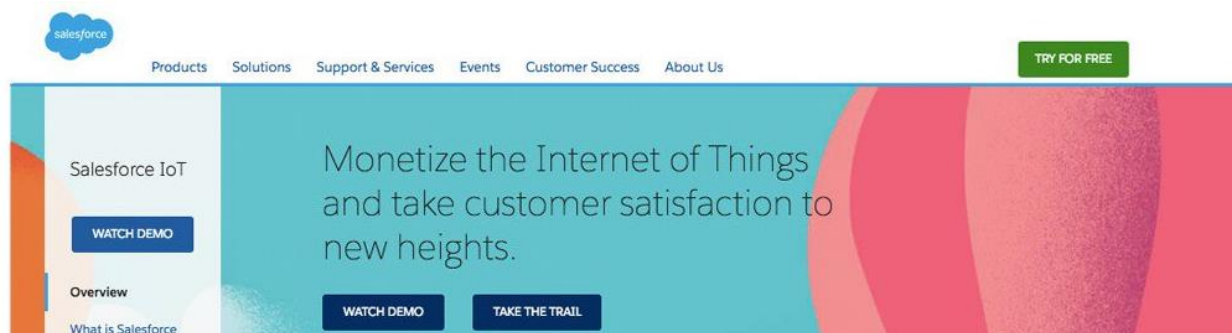


Рисунок 1.7 – Платформа Salesforce IoT

За допомогою Salesforce IoT можливо створювати власні додатки IoT, підключати будь-який пристрій і інтерпретувати його дані для подальшого використання.

Платформа **Oracle Internet of Things** (рис. 1.8) поєднує програмне забезпечення підприємства з "реальним світом" пристроїв і їх метрик. Oracle надає виняткові можливості для бізнесу завдяки гнучкому середовищі для створення комерційних додатків. Будучи визнаним лідером в секторі управління базами даних, Oracle підтримує обробку надзвичайно великого обсягу даних, що дозволяє створювати широкомасштабні мережі IoT. Також, варто згадати використання сучасних механізмів безпеки, які захищають системи IoT від зовнішніх загроз. Оскільки такі системи зазвичай містять різні пристрої, деякі з яких не мають вбудованих засобів безпеки, застосування централізованих заходів безпеки більш ніж виправдано.

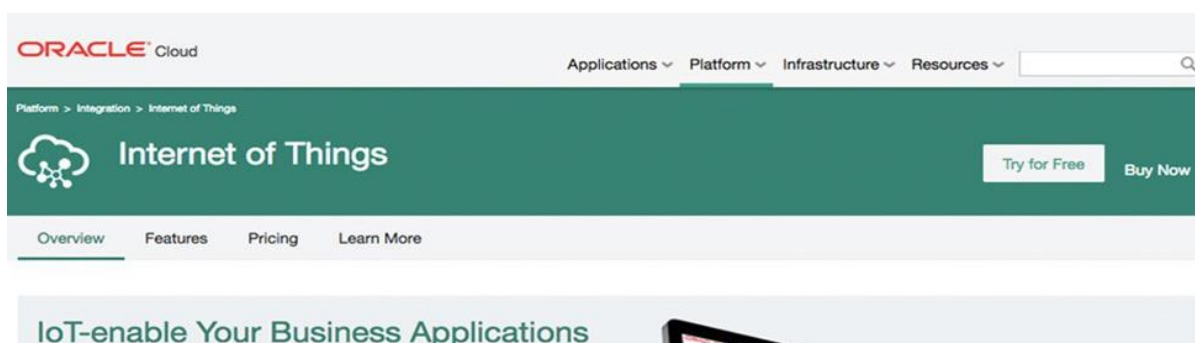


Рисунок 1.8 – Платформа Oracle Internet of Things

Компанія Cisco фокусується на створенні зручної платформи для мобільних IoT-рішень на основі хмарних технологій. Сервіс Cisco підтримує передачу голосу і даних, налаштування додатків IoT та можливості монетизації.

Вибравши платформу **Cisco** (рис. 1.9) для розміщення IoT-додатка, можливо отримати повний пакет функцій управління і моніторингу пристроїв та розширених заходів безпеки – з головним акцентом на мобільний додаток і взаємодію з користувачем. Ряд додаткових послуг дозволяє реалізувати інші функції – наприклад, IoT Services for Utility Networks можна використовувати для створення систем, спеціально призначених для використання комунальними компаніями, а IoT Advisory надає доступ до експертних консультацій за основними бізнес завданням в IoT-секторі.



Рисунок 1.9 – Платформа Cisco Internet of Things

При розробці своєї платформи IoT Bosch використовує відкриті стандарти і відкритий вихідний код та враховує основні вимоги проектів, що містять підключені пристрої і пов'язані технології.

**Bosch IoT Suite** (рис. 1.10) підтримує повний цикл розробки додатків від розробки прототипу до розгортання і обслуговування додатків. Рішення Bosch IoT можуть бути реалізовані як у вигляді хмарних сервісів, так і у вигляді автономних систем, що працюють локально. Платформа підтримує міждоменні додатки, розширюючи можливості інтеграції і зводячи до мінімуму проблеми сумісності.



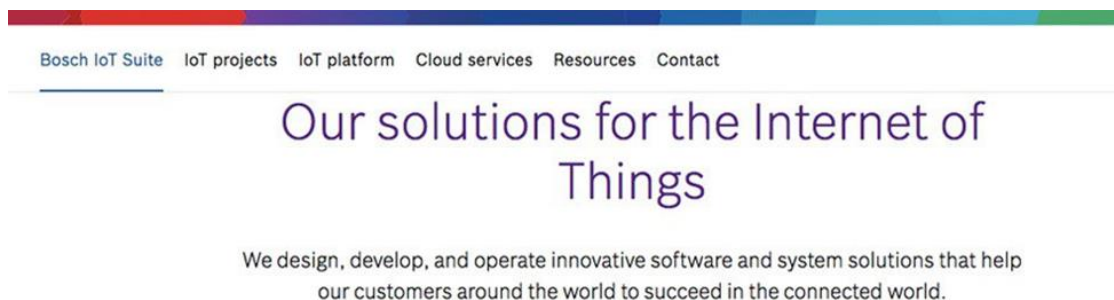


Рисунок 1.10 – Платформа Bosch IoT Suite

Платформа **IBM Watson Internet of Things** (рис. 1.11) підтримує ефективне віддалене управління пристроями, безпечну передачу та зберігання даних в хмарі, обмін даними в режимі реального часу, а також можливості машинного навчання завдяки інтеграції з технологією AI.

Платформа розробки, пропонована IBM, включає в себе ряд зручних інструментів і сервісів, які роблять створення програмного забезпечення IoT більш простим і ефективним.

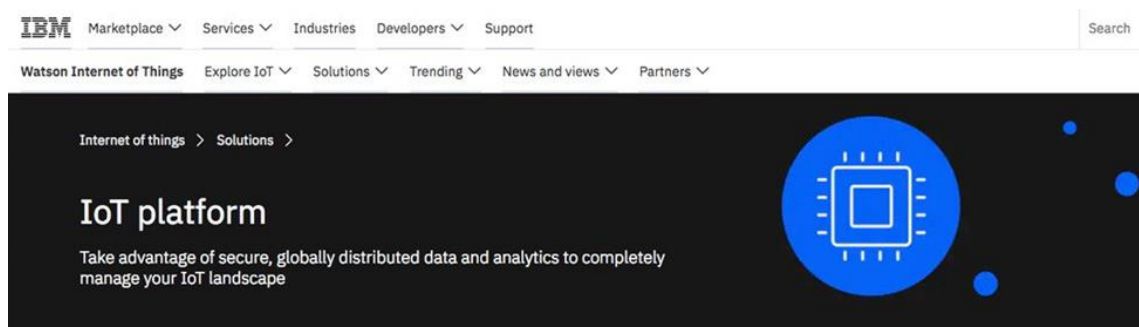


Рисунок 1.11 – Платформа IBM Watson Internet of Things

Платформа ThingWorx від PTC призначена для створення промислових рішень IoT та вважається одним з найбільш повних наборів інструментів для створення додатків IoT різної складності та масштабу.

**ThingWorx IoT Platform** має відмінні можливості для спільного використання і спільної роботи, що робить її відмінним рішенням для великих



груп розробників. Можливостей платформи досить для створення різних додатків IoT без необхідності застосування сторонніх компонентів або бібліотек.

Додатки IoT, створені на основі платформи ThingWorx, володіють всіма функціями вдосконаленого корпоративного рішення – широкими можливостями масштабування і інтеграцією з передовими технологіями, такими як доповнена реальність, і великої аналітикою. Ця потужна функціональність реалізована за допомогою простого та інтуїтивно зрозумілого для користувача інтерфейсу (рис. 1.12), що поєднує високу продуктивність і зручність використання [4].

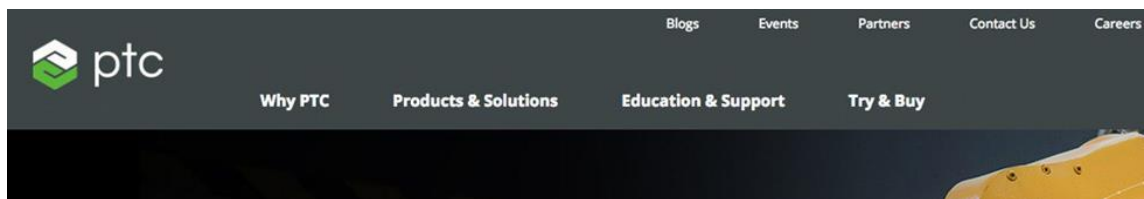


Рисунок 1.12 – Платформа ThingWorx IoT Platform

## РОЗДІЛ 2 ПРОТОКОЛИ ІОТ

Сучасні IoT-рішення будуються на базі протоколів передавання даних (наприклад, MQTT, LoRaWAN та ін.) і забезпечують взаємодію кінцевих пристроїв з хмарними сервісами (наприклад, Azure IoT Suite, Amazon Web Services IoT).

На сьогоднішній день існує декілька протоколів прикладного та канального рівнів використовуваних при створенні IoT-сервісів:

- iBeacon;
- DTLS;
- Eddystone;
- HTTP;
- MQTT;
- PJON;
- LoRaWAN.

Незважаючи на різноманітність протона практики розробники частіше застосовують протоколи MQTT і HTTP. Крім того, їх підтримують основні провайдери хмарних сервісів в своїх рішеннях (Amazon, Microsoft, IBM, Google) [5].

### 2.1 Протокол MQTT

MQTT (Message Queue Telemetry Transport) – це легкий, компактний і відкритий протокол обміну даними створений для передачі даних на віддалених локаціях, де потрібно невеликий розмір коду і є обмеження по пропускній здатності каналу.

Перераховані переваги дозволяють застосовувати протокол MQTT в системах M2M (Machine-to-Machine) та IIoT (Industrial Internet of Things).

MQTT зародився в надрах IBM (розробкою займався Енді Стенфорд-Кларк), а потім був переданий для стандартизації в організацію OASIS (Organization for the Advancement of Structured Information Standards).

В даний час використовується стандарт протоколу версії 3.1. згідно специфікації MQTT V3.1 Protocol Specification, "простий протокол обміну повідомленнями через брокера за схемою публікації/підписки, головна мета якого забезпечити відкритість, простоту, мінімальні вимоги до ресурсів і зручність застосування".

З моменту появи протоколу MQTT мета "зручності впровадження" була досягнута, оскільки було розроблено бібліотеки для впровадження клієнтів MQTT. Посилання майже на всі бібліотеки наведені на сторінці проекту Eclipse Paho.

Також існує версія протоколу MQTT-SN (MQTT for Sensor Networks), раніше відома як MQTT-S, що призначена для вбудованих безпроводових пристроїв без підтримки TCP/IP мереж, наприклад, Zigbee.

Основні особливості протоколу MQTT:

- асинхронний протокол;
- компактні повідомлення;
- робота в умовах нестабільного зв'язку на лінії передавання даних;
- підтримка декількох рівнів якості обслуговування (QoS);
- легка інтеграція нових пристроїв;
- робота на прикладному рівні поверх TCP/IP (рис. 2.1);
- використання порту 1883 при підключенні через SSL.

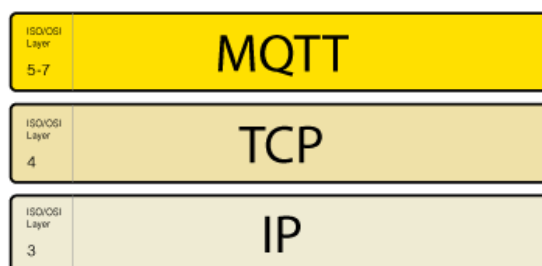


Рисунок 2.1 – Розміщення протоколу MQTT на прикладному рівні

Протокол MQTT не потребує встановлення великої кількості ПЗ. Для клієнта не потрібно встановлювати велику кількість ПО, завдяки чому підходить для пристроїв з обмеженим обсягом пам'яті, таких як Arduino.

Обмін повідомленнями в протоколі MQTT (рис. 2.2) здійснюється між клієнтом (client), який може бути видавцем або підписником (publisher/subscriber) повідомлень, і брокером (broker) повідомлень (наприклад, Mosquitto MQTT).

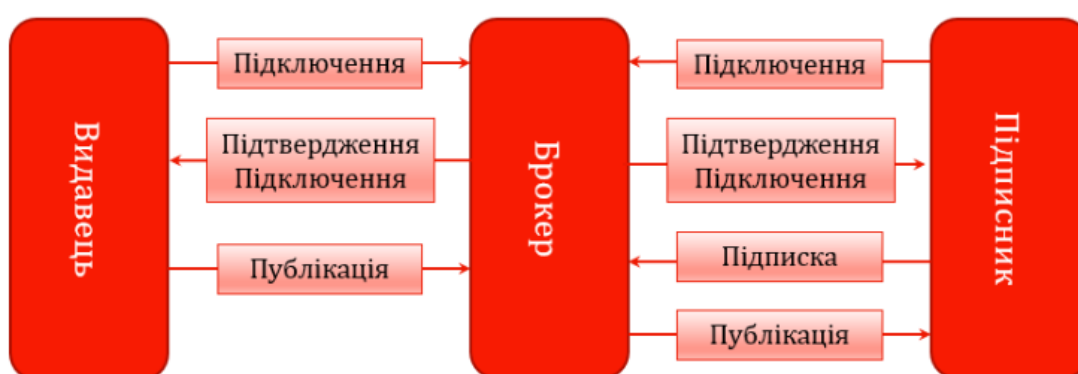


Рисунок 2.2 – Схема обміну повідомленнями в протоколі MQTT

Видавець відправляє дані на MQTT брокер, вказуючи в повідомленні певну тему (topic). Підписники можуть отримувати різні дані від безлічі видавців залежно від підписки на відповідні теми.

Пристрої MQTT використовують різні типи повідомлень для взаємодії з брокером, нижче представлені основні:

- connect – встановити з'єднання з брокером;
- disconnect – розірвати з'єднання з брокером;
- publish – опублікувати дані в темі брокера;
- subscribe – підписатися на тему брокера;
- unsubscribe – відписатися від теми (топіка).

### Семантика програмування топіків

Топіки представляють собою символи з кодуванням UTF-8. Ієрархічна структура топіків має формат "дерева", що спрощує їх організацію та доступ до даних. Топіки складаються з одного або декількох рівнів, які розділені між собою символом "/".

Приклад топіка, в який датчик температури, розташований в спальній кімнаті публікує дані брокеру:

```
/Home/living-space/living-room1/temperature
```

Підписник може так само отримувати дані відразу з декількох топіків, для цього існують wildcard. Вони бувають двох типів: однорівневі і багаторівневі. Для більш простого розуміння розглянемо в прикладах кожен з них.

Для його використання однорівневого wildcard. застосовується символ "+".

Наприклад, нам необхідно отримати дані про температуру у всіх спальних кімнатах:

```
/ Home / living-space / + / temperature
```

В результаті отримуємо дані з топіків:

```
/ Home / living-space / living-room1 / temperature
/ Home / living-space / living-room2 / temperature
/ Home / living-space / living-room3 / temperature
```

Для використання багаторівневого wildcard застосовується символ "#".

Наприклад, щоб отримати дані з різних датчиків всіх спалень в будинку:

```
/ Home / living-space / #
```

В результаті отримуємо дані з топіків:

```
/ Home / living-space / living-room1 / temperature
/ Home / living-space / living-room1 / light1
/ Home / living-space / living-room1 / light2
/ Home / living-space / living-room1 / humidity
/ Home / living-space / living-room2 / temperature
/ Home / living-space / living-room2 / light1
```

## **Структура повідомлень**

MQTT повідомлення складається з декількох частин:

- фіксований заголовок (у всіх повідомленнях);
- змінний заголовок (присутній тільки в певних повідомленнях);
- дані, "навантаження" (присутні тільки в певних повідомленнях)

Структуру фіксованого заголовку наведено на рис. 2.3:

Bit	7	6	5	4	3	2	1	0
Byte 1	Message Type				Flags specific to each MQTT packet			
Byte 2	Remaining Length							

Рисунок 2.3 – Структура фіксованого заголовку

Message Type – це тип повідомлення, наприклад: CONNECT, SUBSCRIBE, PUBLISH та інші.

Flags specific to each MQTT packet – ці 4 біта відведені під допоміжні прапорці, наявність і стан яких залежить від типу повідомлення.

Remaining Length – являє довжину поточного повідомлення (змінний заголовок + дані), може займати від 1 до 4 байтів.

Чотири молодших біта першого байта фіксованого заголовка відведені під спеціальні прапорці (рис. 2.4):

Bit	7	6	5	4	3	2	1	0
Byte 1	Message type				DUP	QoS	QoS	Retain
Byte 2	Remaining Length							

Рисунок 2.4 – Спеціальні прапори фіксованого заголовку

DUP – прапор дубліката встановлюється, коли клієнт або MQTT брокер здійснює повторну відправку пакета (використовується в типах PUBLISH, SUBSCRIBE, UNSUBSCRIBE, PUBREL). При встановленому прапорі змінний заголовок повинен містити Message ID (ідентифікатор повідомлення).

QoS – якість обслуговування (0,1,2).

Retain – при публікації даних з встановленим прапором retain, брокер збереже його. При наступній підписці на цей топик брокер негайно відправить повідомлення з цим прапором. Використовується тільки в повідомленнях з типом PUBLISH.

Всього в протоколі MQTT існує 15 типів повідомлень (Таблиця 2.1).

Таблиця 2.1 – Типи повідомлень в протоколі MQTT

Тип повідомлення	Значення	Напрямок передвання	Опис
Reserved	0000 (0)	Не має	Зарезервовано
CONNECT	0001 (1)	К* -> С**	Запит клієнта на підключення до сервера
CONNACK	0010 (2)	К <- С	Підтвердження успішного підключення
PUBLISH	0011 (3)	К <- С, К -> С	Публікація повідомлення
PUBACK	0100 (4)	К <- С, К -> С	Підтвердження публікації
PUBREC	0101 (5)	К <- С, К -> С	Публікація отримана
PUBREL	0110 (6)	К <- С, К -> С	Дозвіл на видалення повідомлення
PUBCOMP	0111 (7)	К <- С, К -> С	Публікацію завершено
SUBSCRIBE	1000 (8)	К -> С	Запит на підписку
SUBACK	1001 (9)	К <- С	Запит на підписку прийнято
UNSUBSCRIBE	1010 (10)	К -> С	Запит на відписку
UNSUBACK	1011 (11)	К <- С	Запит на відписку прийнято
PINGREQ	1100 (12)	К -> С	PING запит
PINGRESP	1101 (13)	К <- С	PING відповідь
DISCONNECT	1110 (14)	К -> С	Повідомлення про відключення від серверу
Reserved	1111 (15)		Зарезервовано

\* К - клієнт, \*\* С - сервер

Змінний заголовок міститься лише в деяких повідомленнях (рис.2.5)

Bit	7	6	5	4	3	2	1	0
Byte 8	User name	Password	Will Retain	Will QoS		Will Flag	Clean Session	Reserved

Рисунок 2.5 – Структура змінного заголовку

У ньому містяться такі дані:

Packet identifier – ідентифікатор пакета, присутній у всіх типах повідомлень, крім: CONNECT, CONNACK, PUBLISH (з QoS <1), PINGREQ, PINGRESP, DISCONNECT;

Protocol name – назва протоколу (тільки в повідомленнях типу CONNECT);

Protocol version – версія протоколу (тільки в повідомленнях типу CONNECT);

Connect flags – прапори вказують на поведінку клієнта при підключенні.

User name – при наявності цього прапора в «навантаження» має бути вказано ім'я користувача (використовується для аутентифікації клієнта).

Password – при наявності цього прапора в «навантаження» повинен бути вказаний пароль (використовується для аутентифікації клієнта).

Will Retain – при установці 1, брокер зберігає у себе Will Message.

Will QoS – якість обслуговування для Will Message, при встановленому прапорі Will Flag, Will QoS і Will retain є обов'язковими.

Will Flag – при встановленому прапорі, після того, як клієнт відключиться від брокера без відправлення команди DISCONNECT (у випадках непередбачуваного обриву зв'язку і т.д.), брокер сповістить про це всіх підключених до нього клієнтів через так званий Will Message.

Clean Session – очистити сесію. При встановленому "0" брокер збереже сесію, всі підписки клієнта, а також передасть йому всі повідомлення з QoS1 і QoS2, які були отримані брокером під час відключення клієнта, при його наступному підключенні. Відповідно при встановленій "1", під час наступного з'єднання клієнту буде необхідно знову підписуватися на топіки.



Session Present – застосовується в повідомленні з типом CONNACK. Якщо брокер приймає підключення з Clean Session = 1 він повинен встановити "0" в біт Session Present (SP). Якщо брокер приймає підключення з Clean Session = 0, то значення біта SP залежить від того, зберігав чи брокер раніше сесію з цим клієнтом (якщо так, то в SP виставляється 1 і навпаки). Тобто цей параметр дозволяє клієнту визначити чи була збережена брокером попередня сесія.

Connect Return code – якщо брокер з якихось причин не може прийняти правильно сформований CONNECT пакет від клієнта, то в другому байті CONNACK пакета він повинен встановити відповідне значення з нижче зазначеної Таблиці 2.2.

Таблиця 2.2 – Таблиця значень поля Connect Return Code

Значення	Повернене значення	Опис
<b>0</b>	0x00 Connection Accepted	Підключення прийнято
<b>1</b>	0x01 Connection Refused, unacceptable protocol version	Брокер не підтримує версію протоколу, використовувану клієнтом
<b>2</b>	0x02 Connection Refused, identifier rejected	Client ID немає в списку дозволених
<b>3</b>	0x03 Connection Refused, Server unavailable	З'єднання встановлено, але MQTT сервіс не доступний
<b>4</b>	0x04 Connection Refused, bad user name or password	Не правильний логін або пароль
<b>5</b>	0x05 Connection Refused, not authorized	Доступ до підключення заборонений
<b>6-255</b>		Зарезервовано

Зміст і формат даних, переданих в MQTT повідомленнях, визначаються в додатку. Розмір даних може бути обчислений шляхом вирахування з довжини змінного заголовка (Remaining Length).

### **Захист та якість обслуговування в протоколі MQTT (QoS)**

Захист даних в протоколі MQTT передбачає:

- аутентифікація клієнтів – пакет CONNECT може містити в собі поля USERNAME і PASSWORD, при реалізації брокера можна використовувати ці поля для аутентифікації клієнта;
- контроль доступу клієнтів через Client ID;
- підключення до брокера через TLS/SSL.

MQTT підтримує три рівні якості обслуговування (QoS) при передаванні повідомлень.

**QoS 0 At most once.** На цьому рівні видавець один раз відправляє повідомлення брокеру і не чекає підтвердження від нього, тобто відправив і забув (рис. 2.6).



Рисунок 2.6 – Обмін повідомлення при першому рівні обслуговування

**QoS 1 At least once.** Цей рівень гарантує, що повідомлення точно буде доставлено брокеру, але є ймовірність дублювання повідомлень від видавця. Після отримання дубліката повідомлення, брокер знову розсилає тим, хто підписався, а видавцеві знову відправляє підтвердження про отримання повідомлення (рис. 2.7). Якщо видавець не отримав PUBACK повідомлення від брокера, він повторно відправляє цей пакет, при цьому в DUP встановлюється "1".

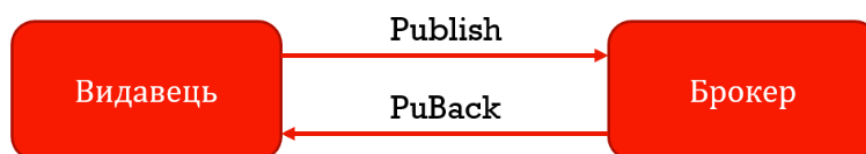


Рисунок 2.7 – Обмін повідомлення при другому рівні обслуговування

**QoS 2 Exactly once.** На цьому рівні гарантується доставка повідомлень підписнику і виключається можливе дублювання відправлених повідомлень (рис. 2.8)

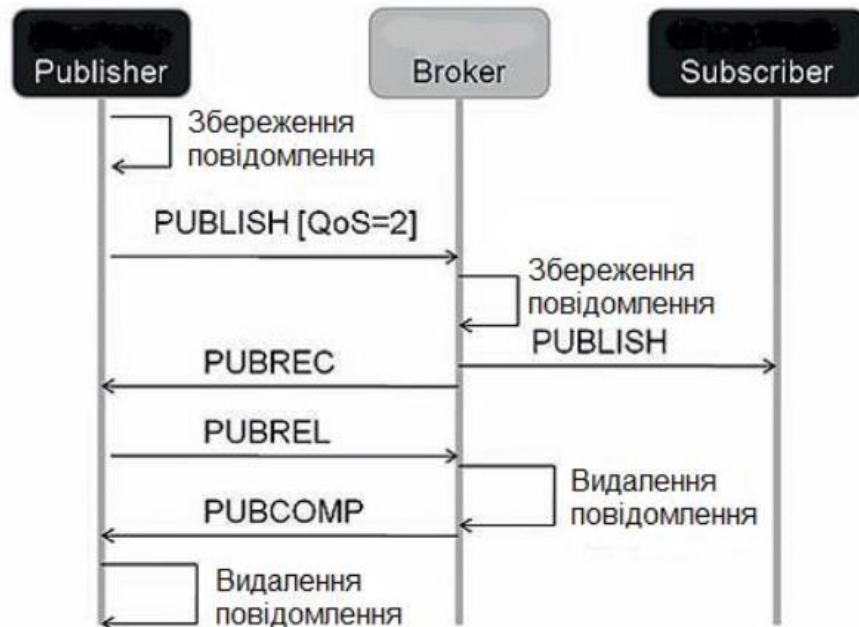


Рисунок 2.8 – Обмін повідомлення при другому рівні обслуговування

Видавець відправляє повідомлення брокеру. У цьому повідомленні вказується унікальний Packet ID, QoS = 2 і DUP = 0. Видавець зберігає повідомлення непідтвердженими поки не отримає від брокера відповідь PUBREC. Брокер відповідає повідомленням PUBREC, в якому міститься той же Packet ID. Після його отримання видавець відправляє PUBREL з тим же Packet ID. До того, як брокер отримає PUBREL він повинен зберігати копію повідомлення у себе. Після отримання PUBREL він видаляє копію повідомлення і відправляє видавцеві повідомлення PUBCOMP про те, що транзакція завершена [6].

### Устаткування для IoT з підтримкою MQTT

Для побудови систем промислового інтернету речей, а також інтегрування існуючого обладнання в хмарні системи використовуються спеціальні пристрої (рис. 2.9):

- web-програмовані PoT контролери;
- PoT модулі віддаленого вводу / виводу;
- шлюзи PoT;
- PoT платформи, сервера MQTT;
- вимірювачі CO, CO<sub>2</sub>, температури і вологості;

## 2.2 Протокол BLE

Завдяки відносно високим швидкостям передавання даних і непоганим енергетичними показниками технологія Bluetooth набула широкого поширення в мобільних електронних пристроях, персональних комп'ютерах, ноутбуках, бездротових навушниках, гарнітурі, мультимедійних центрах. Стандарт дозволяє підтримувати досить розгалужену і складну мережу пристроїв. Однак, для застосування в сенсорних мережах використовувати класичний Bluetooth недоцільно через значне для автономних джерел живлення енергоспоживання внаслідок особливостей роботи стека протоколів.

Bluetooth Low Energy (BLE) є частиною специфікації Bluetooth 4.0, яка також містить протоколи класичного Bluetooth і протокол високошвидкісного Bluetooth (Classic Bluetooth and Bluetooth High Speed Protocols). У порівнянні з класичним Bluetooth, BLE призначений для використання меншої потужності при збереженні аналогічного діапазону зв'язку.

Протокол BLE передбачає передавання даних короткими пакетами за необхідності, потім – вимкнення передавача. На відміну від класичного Bluetooth, пристрої BLE зв'язуються один з одним лише при необхідності відправки або отримання інформації. Це значно зменшує енергоспоживання, що робить його ідеальним для використання в тих випадках, коли потрібне постійне довгострокове з'єднання з низькою швидкістю передавання даних.

На відміну від технологій сенсорних мереж, таких як, ZigBee, 6LoWPAN або Z-Wave, орієнтованих на розгалужені розподілені мережі з численними

передачами даних між вузлами мережі, Bluetooth Low Energy розрахований на топології типу «точка-точка» і «зірка».

BLE ідеально підходить для пульта дистанційного керування телевізором, однак не для безпроводового пристрою потокового передавання мультимедіа, якому для передавання потрібен великий обсяг даних.

Основними областями застосування BLE є пристрої забезпечення безпеки, управління електроприладами і відображення показань, датчики на батареях, спортивні тренажери.

Bluetooth Low Energy вбудований в багато гаджетів, які ми використовуємо сьогодні. Від смартфонів, розумних телевізорів, передових технологій, таких як медичне обладнання, до базових пристроїв, таких як кавомашини, – всі використовують BLE.

Спочатку Nokia розробила BLE для власного проекту під назвою "WIBREE", який згодом було передано спеціальній групі по інтересам Bluetooth (Bluetooth Special Interest Group (SIG)). BLE був задуманий з акцентом на кращу швидкість з'єднання і енергоефективність.

### **Особливості BLE**

#### **1. Наднизьке енергоспоживання**

Для зниження споживання енергії пристрій BLE більшу частину часу проводить в режимі сну. Коли відбувається якась подія, пристрій прокидається і передає коротке повідомлення на шлюз, персональний комп'ютер або смартфон. Максимальне/пікове споживання потужності становить менше 15 мА, а середнє – близько 1 мкА. У порівнянні з класичним Bluetooth, активна споживана потужність знижена в десять разів. У додатках з рідкісною періодичністю включення одна дискова батарея може забезпечити надійну роботу протягом 5-10 років.

#### **2. Малі витрати та сумісність**

Для сумісності з класичною технологією Bluetooth і невисокої ціни реалізації в невеликих пристроях з батарейним типом живлення існують два типи чипсетів:

- дворежимні, з підтримкою функціональності як BLE, так і класичного Bluetooth;
- автономні BLE, оптимізовані для невеликих пристроїв з батарейним живленням, з акцентом на низьку вартість і малу споживану потужність.

### 3. Захищеність, безпека та надійність

У BLE використовується та ж технологія адаптивної стрибкоподібної перебудови частоти (AFH), що і в класичній Bluetooth. Це дозволяє BLE забезпечувати надійне передавання в умовах "зашумленого" ефіру, типових для домашніх, промислових і медичних програм. Для мінімізації витрат і споживання енергії при використанні AFH кількість каналів в BLE скорочено до 40 при ширині кожного каналу 2 МГц, замість 79 каналів шириною 1 МГц, використовуваних в класичній технології Bluetooth.

### 4. Співіснування безпроводових стандартів

Частоту 2.4 ГГц неліцензованого діапазону ISM використовує технологія Bluetooth, безпроводові локальні мережі, IEEE 802.15.4/ZigBee, а також кілька фірмових стандартів. За такої великої кількості технологій в одному радіопросторі завади можуть погіршити характеристики безпроводової мережі (тобто, збільшити затримки і зменшити пропускну здатність) внаслідок необхідності виправлення помилок і повторних передавань. У відповідальних додатках вплив перешкод може бути знижено за рахунок частотного планування та спеціальної конструкції антени. Оскільки і в класичній Bluetooth, і в BLE використовується AFH, що мінімізує завади від інших стандартів радіозв'язку, обмін через Bluetooth стійкий і надійний.

### 5. Дальність зв'язку

Використовувана в BLE технологія модуляції дещо відрізняється від технології класичної Bluetooth. Ця різниця в модуляції забезпечує дальність зв'язку до 300 метрів при потужності передавача радіо-чипсета 10 дБм (максимум, дозволений для BLE).

## Технічні характеристики

### 1. Обмін даними

BLE підтримує дуже короткі пакети даних (від 8 октетів мінімум до 27 октетів максимум), які передаються на швидкості 1 Мбіт/с. Всі з'єднання використовують вдосконалену технологію енергозбереження для максимального скорочення робочого циклу з метою мінімізації споживання енергії.

### 2. Стрибоподібна перебудова частоти

Для зниження завад від засобів радіозв'язку інших стандартів, що працюють в ISM діапазоні 2.4 ГГц, BLE використовує загальну для всіх версій Bluetooth технологію AFH. Переваги багатотрактової маршрутизації збільшують енергетичний бюджет каналу зв'язку і ефективний робочий діапазон, а також оптимізують споживання енергії.

### 3. Управління хостом

Значна частина інтелекту BLE реалізується контролером. Це дозволяє ведучому пристрою довше залишатися в стані сну і прокидатися по сигналу контролера тільки тоді, коли хост повинен виконати будь-яку дію. Відповідно, скорочується споживання струму, оскільки хост процесор, як правило, споживає більше енергії, ніж контролер BLE.

### 4. Затримка

BLE може підтримувати швидке встановлення з'єднання і передавання даних за 3 мс. Це дозволяє додатку всього за кілька мілісекунд встановити з'єднання і відправити аутентифіковані дані, а потім швидко розірвати з'єднання.

### 5. Дальність зв'язку

Збільшення індексу модуляції дозволило забезпечити для BLE максимальну дальність зв'язку більше 100 метрів.

### 6. Надійність

Для забезпечення максимальної завадостійкості BLE контролює всі пакети за допомогою надійного 24-бітного алгоритму CRC.

## 7. Високий рівень безпеки

Повний алгоритм AES-128, який використовує блоковий протокол CCM, забезпечує надійне шифрування і аутентифікацію пакетів даних, що гарантують безпеку інформаційного обміну.

## 8. Топологія

BLE використовує 32-бітову адресу звернення на кожен пакет для кожного відомого пристрою, дозволяючи підключати мільярди пристроїв. Технологія оптимізована для з'єднань точка-точка, при цьому допускаються багатоточкові сполуки з використанням зіркоподібної топології.

### **Основні поняття в BLE**

У BLE є два основних поняття:

- GAP – Generic Access Profile (загальний профіль доступу);
- GATT – Generic Attribute Protocol (протокол загальних атрибутів).

#### **Загальний профіль доступу (GAP)**

GAP відповідальний за підключення і поширення інформації про наявність пристрою BLE. GAP відповідає за видимість пристрою в зовнішньому світі, а також відіграє важливу роль у визначенні того, як пристрій взаємодіє з іншими пристроями.

Наступні дві концепції є невід'ємною частиною GAP:

- периферійні пристрої;
- центральні пристрої.

Периферійні пристрої це невеликі пристрої з низьким енергоспоживанням, які можуть підключатися до складних, більш потужним центральним пристроям. Монітор серцевого ритму є прикладом периферійного пристрою.

Центральні пристрої це в основному мобільні телефони або гаджети зі збільшеною пам'яттю і обчислювальною потужністю.

#### **Протокол загальних атрибутів (GATT)**

Використовуючи загальний протокол даних, відомий як протокол атрибутів, GATT визначає, як два пристрої BLE обмінюються даними один з одним,



використовуючи поняття – сервіс (service) і характеристика (characteristic). Цей протокол зберігає всі сервіси та характеристики в довідковій таблиці з використанням 16-бітних ідентифікаторів, як зазначено в Bluetooth SIG. Важливо відзначити, що GATT ініціюється тільки після того, як Advertising процес, регульований GAP, завершений.

Дві основні концепції, які утворюють GATT

- сервіси (service);
- характеристики (characteristic).

Сервіси можна уявити як шафу, в якій може бути багато ящиків, що в свою чергу називаються характеристиками. Сервіс може мати багато характеристик. Кожен сервіс унікальний сам по собі з універсально унікальним ідентифікатором (UUID), який може бути розміром 16 біт для офіційних адаптованих сервісів або 128 біт для призначених для користувача сервісів.

Advertising process (забезпечення видимості пристрою)

Процес виявлення пристроїв полягає в тому, що периферійний пристрій в задані інтервали відправляє в округ дані про своє існування. Якщо ці дані отримає центральний пристрій, то він відправить запит на сканування. У відповідь периферійний пристрій надішле дані результату сканування.

Периферійний пристрій буде відправляти “рекламні” дані кожні 2 секунди. Якщо центральний пристрій готовий прослухати рекламні пакети, він відповість запитом сканування. У відповідь на цей запит периферійний пристрій відправить дані відповіді сканування. Таким чином, центральний і периферійний пристрої дізнаються і зв’язується один з одним [7].

### **Структура стека протоколів Bluetooth Low Energy**

Як і класичний стек протоколів Bluetooth, стек BLE складається з двох основних частин: контролера (controller) і вузла мережі (host). Контролер включає в себе фізичний і канальний рівень і часто реалізується у вигляді системи-на-кристалі (СНК) з інтегрованим безпроводовим трансивером. Частина стека, що називається вузлом мережі реалізується програмно на

мікроконтролері додатків і включає в себе функціональність верхніх рівнів: рівень логічного зв'язку (Logical Link Control – LLC), протокол адаптації (Adaptation Protocol – L2CAP), протокол атрибутів (Attribute Protocol – ATT), протокол атрибутів профілів пристроїв (Generic Attribute Profile – GATT), протокол забезпечення безпеки (Security Manager Protocol – SMP), протокол забезпечення доступу до функцій профілю пристроїв (Generic Access Profile (GAP)). Взаємодія між верхньою і нижньою частинами стека здійснюється інтерфейсом Host Controller Interface (HCI). Додаткова функціональність прикладного рівня може бути реалізована поверх рівня вузла мережі.

На рис. 2.9 представлена структура стека протоколів BLE та на рис. 2.10 структура пакета даних BLE.

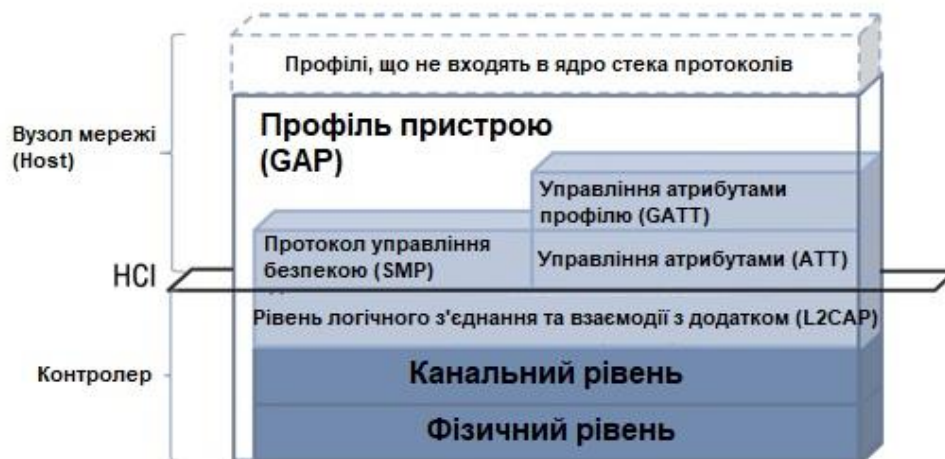


Рисунок 2.9 – Структура стека протоколів BLE

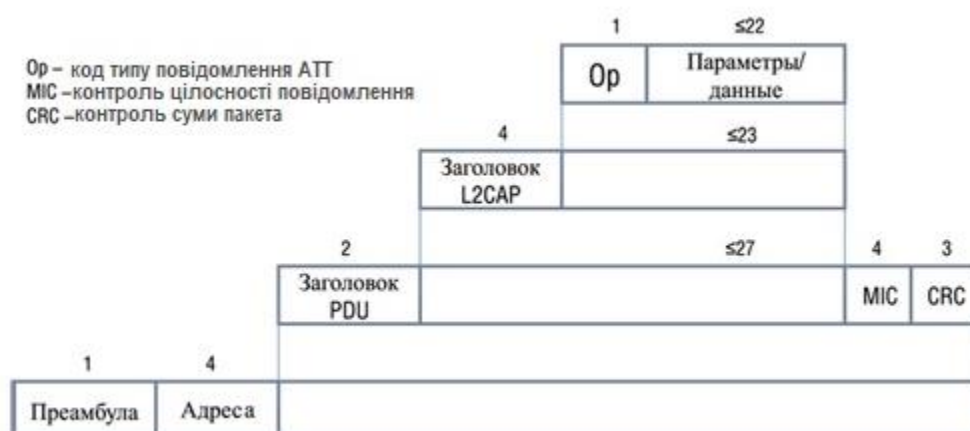


Рисунок 2.10 – Структура стека протоколів BLE

## Модулі Bluetooth Low Energy

Протокол BLE міститься в модулях з вбудованим програмним забезпеченням. Модулями оснащуються кінцеві пристрої.

Як приклад, розглянемо модулі компанії BlueGiga:

- BT111
- BLE112
- USB BLED112

BT111 призначений для додатків, в яких необхідна робота і з класичними Bluetooth-пристроями, і з пристроями Bluetooth Low Energy, і являє собою мініатюрний модуль поверхневого монтажу з вбудованою антеною (рис. 2.11).



Рисунок 2.11 – Зовнішній вигляд дворежимного BLE-модуля BT111

Модуль BLE112 (рис. 2.12) є однорежимним BLE-модулем, що призначений для сенсорних систем і BLE-аксесуарів на батарейках. BLE112 підтримує практично всі можливості пристроїв BLE – безпроводове передавання даних, підтримка стека протоколів BLE і ряду профілів BLE-пристроїв, додатково присутня можливість зберігання призначених для користувача додатків. Таким чином, можлива робота модуля BLE без зовнішнього контролера.



Рисунок 2.12 – Зовнішній вигляд модуля BLE112

Ще одним пристроєм, для додатків BLE, є USB-BLE модуль BLED112 (рис. 2.13). Зберігаючи функціональність, аналогічну модулю BLE112 (за винятком можливостей вводу-виводу), він виконаний у форматі USB-пристрою і дозволяє підключати інші BLE до персонального комп'ютера. BLED112 може також виконувати роль віртуального COM-порту або USB-HID пристрою [8].



Рисунок 2.13 – Зовнішній вигляд USB-BLE модуля BLED112

### 2.3 Протокол LoRaWAN

Вперше мережевий протокол LoRaWAN (Long Range Wide Area Networks) було презентовано на ринку безпроводових технологій в 2015 році.

LoRaWAN – це двонаправлений протокол, який використовує всі переваги технології LoRa для надання послуг, включаючи надійну доставку повідомлень, наскрізну безпеку, визначення місця розташування і можливості під LGPL.

Semtech Corporation і дослідницький центр IBM Research представили новий відкритий мережевий протокол, при цьому високо оцінивши його конкурентоспроможність в порівнянні з Wi-Fi і стільниковими мережами. Також був відзначений ряд переваг LoRaWAN – це, в першу чергу, можливість розгортання міжмашинних (M2M) комунікацій і звичайно максимальна енергоефективність нової технології.

Поява технології LoRaWAN викликала великий резонанс на ринку безпроводового зв'язку, що спричинило за собою необхідність прийняти єдиний стандарт для глобальних мереж з низьким енергоспоживанням – LPWAN (з англ. Low Power Wide Area Network). Власне, аббревіатура LoRa поєднує в собі метод модуляції LoRa в безпроводових мережах LPWAN, розроблений Semtech, і відкритий протокол LoRaWAN.

Для підтримки, розвитку та стандартизації нової технології була створена некомерційна організація LoRa Alliance, її засновниками стали найбільші відомі виробники електроніки (IBM, Semtech, Cisco, Kerlink, IMST і ін.) та провідні телекомунікаційні оператори (Bouygues Telecom, KPN, SingTel, Proximus, Swisscom). Введення стандарту дозволить об'єднати мільйони пристроїв в IoT максимально просто, а також вирішити проблему надання послуг IoT організаціям і фізичним особам через операторів зв'язку.

У той час як інші технології бездротового зв'язку, такі як Bluetooth і BLE (а в деякій мірі Wi-Fi і ZigBee), неможливо встановити для передавання даних на великі відстані, LPWAN, і зокрема її реалізація – LoRaWAN, навпаки, забезпечує обмін невеликими обсягами даних на значні дистанції.

Технологія, використовувана в мережі LoRaWAN, призначена для підключення недорогих датчиків на батарейках, що працюють на великих відстанях в жорстких умовах, які раніше були занадто складними або дорогими для підключення. При цьому витрати електроенергії мінімальні, що дозволяє досягати максимального часу автономної роботи (кілька років) на одному акумуляторі типу AA.

Завдяки значній проникній здатності сигналу, шлюз LoRaWAN, розміщений на будівлі або вежі, може підключатися до датчиків або лічильників, розташованих більш ніж в 16 кілометрах.

### Архітектура мережі LoRaWAN

На відміну від топології стільникової мережі, прийнятої в більшості мереж, LoRaWAN використовує архітектуру мережі "зірка", тому замість того, щоб кожен кінцевий пристрій було майже завжди включено, повторюючи передавання від інших пристроїв для збільшення дальності, кінцеві пристрої в мережі LoRaWAN зв'язуються безпосередньо з шлюзами і працюють тільки тоді, коли їм потрібно зв'язатися зі шлюзом, оскільки дальність зв'язку не є проблемою. Це є чинником, що сприяє низькому енергоспоживанню і високому часу автономної роботи, отриманим в кінцевих пристроях LoRa.

Архітектура мережі LoRa (рис. 2.14) складається з чотирьох основних частин:

- кінцеві пристрої;
- шлюзи;
- мережевий сервер;
- сервер додатків.

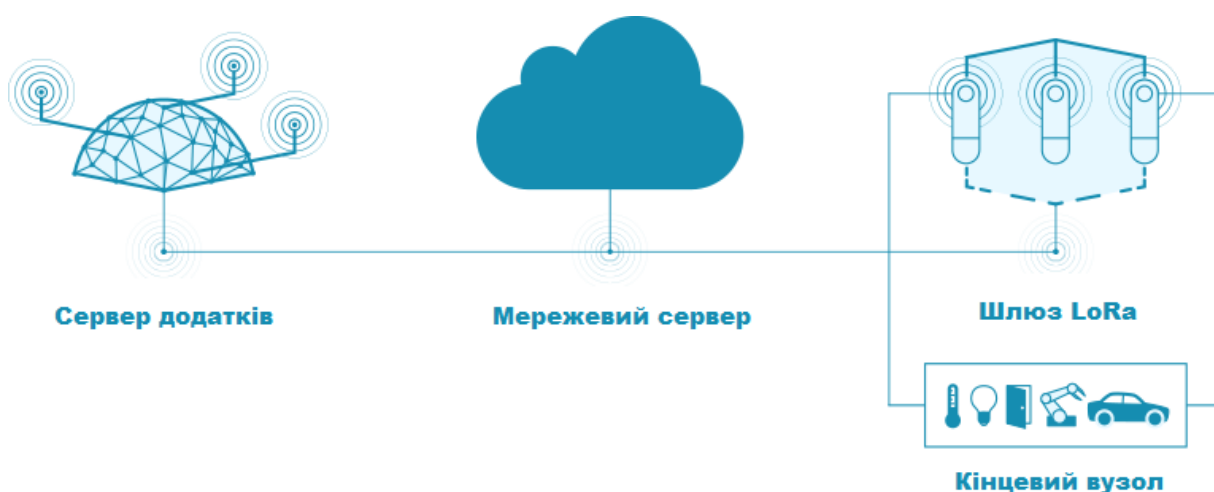


Рисунок 2.14 – Архітектура мережі LoRa

**Кінцеві пристрої** – це датчики або виконавчі механізми на межі мережі. Кінцеві пристрої обслуговують різні додатки і мають свої вимоги. Для оптимізації різних профілів кінцевих додатків LoRaWAN використовує три різних класи пристроїв, для яких можуть бути налаштовані кінцеві пристрої. Класи показують компроміси між затримкою зв'язку по низхідній лінії зв'язку і часом автономної роботи пристрою. Три основні класи:

- двонаправлені кінцеві пристрої (клас А);
- двонаправлені кінцеві пристрої з запланованими слотами прийому (клас В);
- двонаправлені кінцеві пристрої з максимальними слотами прийому (клас С).

Двонаправлені кінцеві пристрої "класу А" (Bi-directional end-devices, Class A). Подібні пристрої застосовуються, коли потрібно підтримувати мінімальну потужність при перевазі передавання даних до сервера. Кінцевий вузол ініціює сеанс зв'язку шляхом відправки пакета даних, після чого виділяє два вікна, в перебігу яких чекає даних від сервера. Відповідно, можливість передавання даних між сервером і кінцевим пристроєм виникає тільки після відкриття сеансу кінцевим пристроєм.

Двонаправлені кінцеві пристрої "класу В" (Bi-directional end-devices, Class B). Відрізняються від пристроїв "класу А", так як мають можливість за розкладом відкривати додаткові вікна прийому. Для складання розкладу кінцевий пристрій здійснює синхронізацію за спеціальним сигналом від шлюзу. Таким чином, наявність додаткового вікна дозволяє серверу обмінюватися даними в попередньо обумовлений момент часу.

Двонаправлені кінцеві пристрої "класу С" з максимальним приймальним вікном (Bi-directional end-devices, Class C). Відрізняються практично безперервним вікном прийому даних і закривають його лише на час передавання даних. Така особливість, дозволяє застосовувати їх для вирішення завдань, пов'язаних з великим об'ємом даних.

**Шлюзи** (концентратори) – це пристрої, підключені до мережевого сервера через стандартні IP-з'єднання, які ретранслюють повідомлення між серверною частиною центральної мережі і кінцевими пристроями, використовуючи протокол безпроводового зв'язку з одним стрибком. Вони призначені для підтримки двостороннього зв'язку і оснащені багатоадресним передаванням, що дозволяє програмному забезпеченню відправляти повідомлення масового поширення, такі як безпроводові оновлення. В основі кожного шлюзу LoRa лежить багатоканальний демодулятор LoRa, здатний декодувати всі варіанти модуляції LoRa на декількох частотах паралельно. Технологія з розширеним спектром використовує швидкості передавання даних в діапазоні від 0,3 кбіт/с до 50 кбіт/с, щоб запобігти взаємодії один з одним, і створює набір "віртуальних" каналів, які збільшують пропускну здатність шлюзу. Щоб максимізувати час автономної роботи кінцевих пристроїв і загальну ємність мережі, мережевий сервер LoRa керує швидкістю передавання даних і РЧ-виходом для кожного кінцевого пристрою окремо за допомогою схеми адаптивної швидкості передавання даних (ADR).

**Мережевий сервер Lora** – це інтерфейс між сервером додатків і шлюзами. Він передає команди від сервера додатків до шлюзу, перенаправляючи дані зі шлюзів на сервер додатків. Він виконує різні функції, включаючи забезпечення відсутності пакетів дублювання, планування підтверджень, управління швидкістю передавання даних і РЧ-виходом для кожного кінцевого пристрою індивідуально з використанням схеми адаптивної швидкості передавання даних (ADR).

**Сервер додатків** визначає, для чого використовуються дані від кінцевих пристроїв. Як сервер додатків найкраще використовувати "хмарні" платформи від провідних гравців ринку, наприклад: AWS IoT, хмарна платформа для IoT, випущена Amazon. Ця інфраструктура дозволяє інтелектуальним пристроям легко підключатися і безпечно взаємодіяти з хмарою AWS і іншими підключеними пристроями. Microsoft Azure IoT Suite, платформа, яка складається з набору служб, що дозволяють користувачам взаємодіяти і



отримувати дані зі своїх пристроїв IoT, а також виконувати різні операції над даними, такі як багатовимірний аналіз, перетворення і агрегування.

### **Безпека та конфіденційність в протоколі LoRaWAN**

Важливість безпеки і конфіденційності в будь-якому рішенні IoT не можна переоцінити. Протокол LoRaWAN визначає шифрування для забезпечення безпеки даних, а саме:

- ключі AES128 для кожного пристрою;
- миттєва регенерація/відгук ключів пристрої;
- пакетне шифрування корисного навантаження для конфіденційності даних;
- захист від повторних атак;
- захист від атак "людина посередині".

LoRa використовує два ключі. Ключі сеансу мережі і сеансу додатка, обидва з яких забезпечують роздільний, зашифрований зв'язок для управління мережею і взаємодії додатків. Ключ сеансу мережі, спільно використовуваний пристроєм і мережею, відповідає за аутентифікацію даних кінцевого вузла, в той час як ключ сеансу додатка, використовуваний додатком і кінцевим вузлом, відповідає за забезпечення конфіденційності даних пристрою [9].

## **2.4 Протокол Ethernet/ IP**

Ethernet/IP – це відкритий промисловий протокол, який підтримує обмін повідомленнями (обмін повідомленнями введення/виведення в реальному часі). Стандарт EtherNet/IP забезпечує об'єднання в єдиний інформаційний простір всіх компонентів систем автоматизації IoT – від рівнів засобів введення/виведення, контролерного обладнання, серверів до рівня систем управління підприємством.

Зростання застосування інтелектуальних пристроїв в системах управління виробничими процесами спричинило за собою розвиток технологій для промислових мереж передавання даних. Технологія Ethernet зі

своєю стандартною архітектурою, як і інші комерційні технології, мала ряд обмежень, які не дозволяли використовувати її на промислових майданчиках, тому виробники обладнання були змушені розробляти свої системи для забезпечення обміну даними. Але технологія Ethernet продовжувала розвиватися, і на початку 1980-х років з'явилися перші розробки на основі шинної технології, які отримали узагальнену назву «промисловий Ethernet». У 1998 році міжнародна група ControlNet SIG запропонувала технологію для обміну даними між мережами Ethernet і ControlNet, DeviceNet, а в 2000 році на її основі організації ControlNet International, Industrial Ethernet Association і Open DeviceNet Vendor Association (ODVA) затвердили новий стандарт – EtherNet/IP (Ethernet Industrial Protocol). Завдяки спільним зусиллям цих організацій в даний час близько двохсот виробників пропонують продукти, що підтримують протокол EtherNet/IP. Членами організації ODVA, що підтримують стандарт EtherNet/IP, є такі великі компанії, як Rockwell Automation, Schneider Electric, Cisco Systems, Omron, та багато інших постачальників обладнання для автоматизації. EtherNet/IP є відкритою мережею, так як використовує стандарт Ethernet IEEE 802.3, набір протоколів TCP/IP, стандартний промисловий протокол (Common Industrial Protocol – CIP), а також інформаційний протокол і протокол введення-виведення в режимі реального часу, які використовують мережі DeviceNet і ControlNet.

EtherNet/IP використовує модель даних “постачальник/споживач” (producer/consumer), яка надає такі переваги перед традиційними моделями “клієнт/сервер” (client/server) та джерело/місце призначення (source/destination):

- різноманітні вузли можуть "споживати" одні і ті ж дані від одного "виробника";
- вузли можуть бути синхронізовані;
- оптимізація пропускнуєї спроможності для кращої продуктивності;
- протокол легко керує даними зі збору, конфігурації та управління через єдину мережу [10];

- для управління мережею не потрібен провідний пристрій

Також EtherNet/IP підтримує наступні типи обміну даними введення/виведення: опитувальний (polled), циклічний (cyclic), груповий (multicast) і обмін даними по зміні стану (change of state).

Модель постачальник/споживач (producer/consumer) є службою незалежною від лінії зв'язку, і підтримує явний і неявний обмін інформаційними повідомленнями введення/виведення одночасно по єдиній шині.

Ідеальними застосуваннями мережі EtherNet/IP є ситуації, де є необхідність поєднання промислових і комерційних технологій в єдину мережу. А також системи, які мають велике число вузлів в мережі, вимагають пряме підключення до бізнес-систем, що необхідно інтегрувати безпосередньо в корпоративну інфраструктуру, і, нарешті, додатки, які вимагають максимальної гнучкості у ставленні до швидкодії, топології і пропускну здатності.

Основні переваги мережі EtherNet/IP:

- мережа EtherNet/IP надає всі можливості CIP
- дозволяє великій кількості інформації, даними по конфігурації і даними введення/виведення "спілкуватися" по одній високошвидкісній мережі;
- дозволяє клієнтам тісно пов'язувати технологічні операції з корпоративними операціями;
- сприяє скороченню витрат з технічного обслуговування завдяки повторному використанню наявних мережевих ресурсів і коштів;
- дозволяє комерційним і промисловим технологічним рівнями існувати в єдиній мережі.

На додаток до перерахованих переваг, мережа EtherNet/IP надає додаткові можливості завдяки високошвидкісному передаванню даних, а також підтримку стандартних мереж і протоколів Ethernet, а саме TCP/IP, HTTP.

## РОЗДІЛ 3 ЛАБОРАТОРНИЙ ПРАКТИКУМ

### 3.1 Лабораторна робота №1. Вивчення роботи датчика тиску BMP180

**Мета:** дослідити роботу датчика тиску BMP180 використовуючи контролер Arduino.

**Завдання:** написати програму для вимірювання атмосферного тиску в залежності від висоти знаходження над рівнем моря.

**Обладнання:** мікроконтролер Arduino; проводи; датчик тиску BMP180; макетна плата; USB – кабель.

#### Загальні відомості

Датчики тиску працюють на перетворенні тиску в рух механічної частини. Датчик тиску складається з перетворювача з чутливим елементом, корпусу, механічних елементів (мембран, пружин) та електронної схеми.

Датчик BMP180 – це недорогий і простий в застосуванні сенсорний датчик, що вимірює атмосферний тиск і температуру. Загальний вигляд датчика BMP180 наведено на рис. 3.1. Зазвичай використовується для визначення висоти і в метеостанціях. Складається пристрій з п'єзореzystивного датчика, термодатчика, АЦП, незалежної пам'яті, ОЗУ і мікроконтролера [11].



Рисунок 3.1 – Датчик тиску і температури BMP180 з роз'ємом

Технічні характеристики датчика BMP180:

- межі вимірюваного тиску 225-825 мм ртутного стопчика;
- напруга живлення 3,3 – 5 В;

- струм 0,5 мА;
- підтримка інтерфейсу I2C;
- час спрацювання 4,5 мс;
- діапазон вимірюваних значень: від 300 гПа до 1100 гПа (від -500 м до + 9000 м над рівнем моря);
- рівень шуму 0.06 гПа (0.5 м) в грубому режимі (ultra low power mode) і 0.02 гПа (0.17 м) в режимі максимального дозволу (advanced resolution mode);
- розміри 15 x 14 мм.

Для підключення землі з Ардуїно потрібно з'єднати з землею на датчику, напруга – на 3,3 В, SDA – до пін A4, SCL – A5. Контакти A4 і A5 обираються з урахуванням їх підтримки інтерфейсу I2C. Сам датчик працює від напруги 3,3 В, а Ардуїно – від 5 В, тому на модулі з датчиком стабілізатор напруги.

Для роботи з датчиком знадобиться бібліотека: BMP180\_Breakout\_Arduino\_Library. Її треба завантажити зі сховища, і встановити в Arduino IDE.

### Приклад коду

```
#include <SFE_BMP180.h>
#include <Wire.h>
SFE_BMP180 pressure;
#define ALTITUDE 0 // Altitude of SparkFun's HQ in Boulder,
CO. in meters
void setup()
{
  Serial.begin(9600);
  Serial.println("REBOOT");
  if (pressure.begin())
    Serial.println("BMP180 init success");
  else
  {
    Serial.println("BMP180 init fail\n\n");
    while(1); // Pause forever.
  }
}

void loop()
{
```

```

char status;
double T,P,p0,a;
Serial.println();
Serial.print("provided altitude: ");
Serial.print(ALTITUDE,0);
Serial.print(" meters, ");
Serial.print(ALTITUDE*3.28084,0);
Serial.println(" feet");
status = pressure.startTemperature();
if (status != 0)
{
    status = pressure.getTemperature(T);
    if (status != 0)
    {
        // Print out the measurement:
        Serial.print("temperature: ");
        Serial.print(T,2);
        Serial.print(" deg C, ");
        Serial.print((9.0/5.0)*T+32.0,2);
        Serial.println(" deg F");
        status = pressure.startPressure(3);
        if (status != 0)
        {
            status = pressure.getPressure(P,T);
            if (status != 0)
            {
                Serial.print("absolute pressure: ");
                Serial.print(P,2);
                Serial.print(" mb, ");
                Serial.print(P*0.0295333727,2);
                Serial.println(" inHg");
                p0 = pressure.sealevel(P,ALTITUDE); // we're at
1655 meters (Boulder, CO)
                Serial.print("relative (sea-level) pressure: ");
                Serial.print(p0,2);
                Serial.print(" mb, ");
                Serial.print(p0*0.0295333727,2);
                Serial.println(" inHg");
                a = pressure.altitude(P,p0);
                Serial.print("computed altitude: ");
                Serial.print(a,0);
                Serial.print(" meters, ");
                Serial.print(a*3.28084,0);
                Serial.println(" feet");
            }
            else Serial.println("error retrieving pressure
measurement\n");
        }
        else Serial.println("error starting pressure
measurement\n");
    }
    else Serial.println("error retrieving temperature
measurement\n");
}

```

```

    }
    else Serial.println("error starting temperature
measurement\n");

    delay(5000); // Pause for 5 seconds.
}

```

### Хід виконання роботи

1. Зібрати макет відповідно смехі (рис. 3.2).
2. Підключити схему до живлення (5 В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.

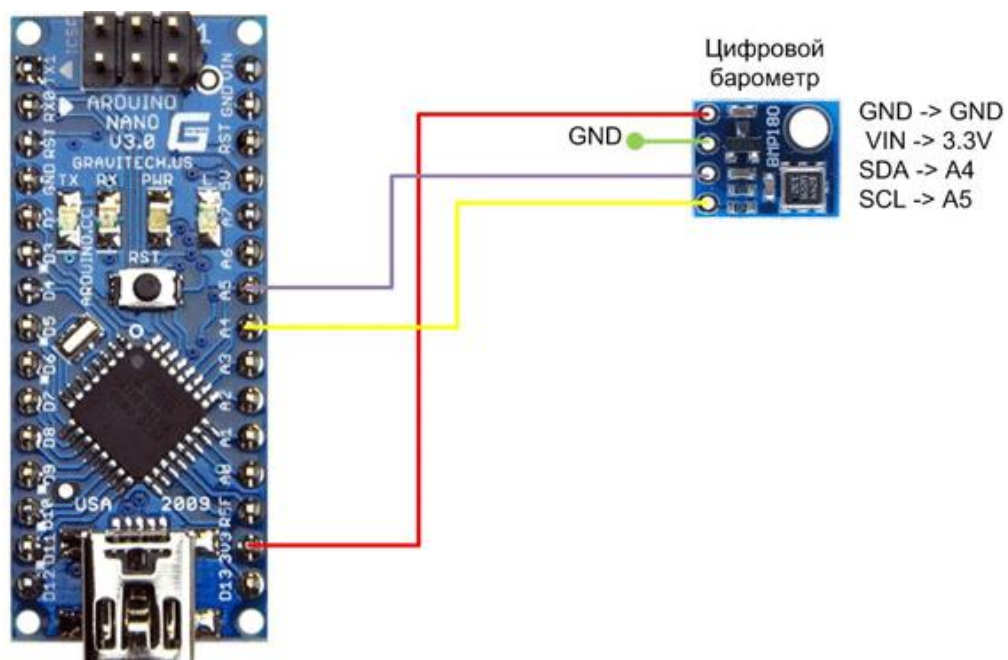


Рисунок 3.2 – Схема макету

### Завдання

Зібрати схему і написати програму, що буде вимірювати атмосферний тиск в залежності від висоти знаходження над рівнем моря.

Налагодити програму в середовищі Arduino і перевірити на макеті.

### Алгоритм роботи програми

1. Запитуємо у барометра тиск.
2. Чекаємо деякий час, поки датчик оцінює тиск.
3. Отримуємо значення тиску;
4. Повертаємо значення тиску з функції.

### **Контрольні питання**

1. В чому полягає принцип роботи датчика BMP180?
2. Назвіть основне місце використання датчиків тиску.
3. Назвіть основні технічні характеристики BMP180.
4. Опишіть основні складові побудови датчика BMP180.
5. Опишіть алгоритм підключення датчика BMP180 до плати Arduino.

**Підготувати** звіт згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).



## 3.2 Лабораторна робота №2. Взаємодія контролера Arduino з програмою Node-RED

**Мета:** дослідження інструмента для візуального програмування потоком даних Node-RED.

**Завдання:** навчитися приймати и передавати дані з комп'ютера в плату Arduino використовуючи програмне середовище Node-RED.

**Обладнання:** мікроконтролер Arduino; проводи; макетна плата; USB-кабель, світлодіоди.

### Загальні відомості

Node-RED – інструмент для візуального програмування потоком даних, розроблений працівниками компанії IBM для поєднання різноманітних пристроїв, API та онлайн-сервісів як складових частин Інтернету речей.

Node-RED дає змогу працювати з браузерним редактором потоків даних як окремими вузлами з різним функціоналом, що уможливорюють створення JavaScript-функцій. Причому можна використовувати як базові вузли, якими одразу забезпечений Node-RED, так і встановлювати вузли з додатковим функціоналом з репозиторію NPM або ж навіть створити свій власний вузол з унікальним функціоналом. Програми або ж їхні частини, розроблені за допомогою Node-RED, можуть бути збережені та поширені для вільного використання. Саме середовище побудовано на основі Node.js. Потоки, створені за допомогою Node-RED, зберігаються у вигляді JSON. Починаючи з MQTT версії 0.14, вузли можна налаштовувати для TLS-з'єднання [12].

### Приклад коду

```
int pin8(8); //Assigning integrer ID to pin #8

void setup() {
    pinMode(pin8,OUTPUT); //Configuring pin #8 to be an output
    pin.
    Serial.begin(9600); // Serial communication at 9600 baud rate.
}

void loop() {
    int SerialData = Serial.read(); //Read Serial data.
```

```
Serial.println(SerialData); // return the recieved Serial
data.

if (SerialData == 54) {
    digitalWrite(pin8,HIGH); //Writing to Digital Pin #8 5 volt
}
else if (SerialData == 55) {
    digitalWrite(pin8,LOW); //Writing to Digital Pin #8 0 volt
}
delay(2000);
}
```

### Хід виконання роботи

1. Встановити Node. js.
2. Зібрати макет відповідно до завдання (рис. 3.6).
3. Підключити схему до живлення (5 В).
4. Завантажити програму в мікроконтролер Arduino.
5. Перевірити правильність роботи програми.

### Завдання

Спочатку потрібно встановити Node. js, для цього перейшовши по даному посиланню <https://nodejs.org/en/>.

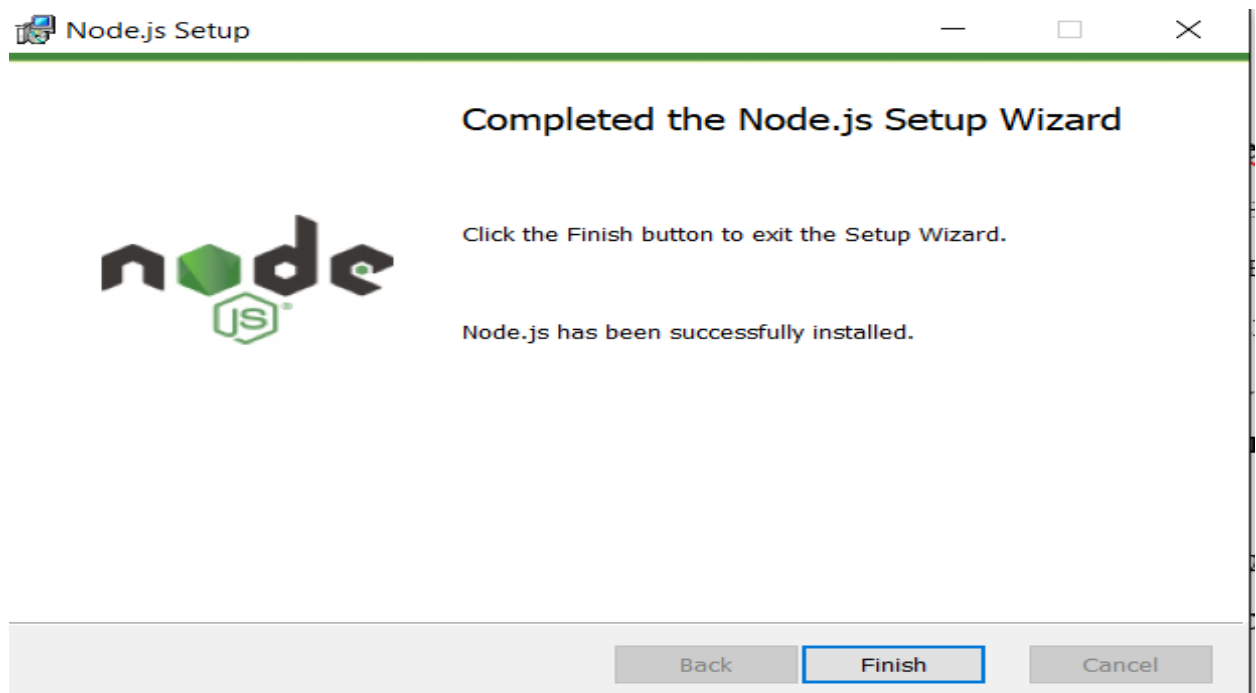


Рисунок 3.3 – Встановлення Node.js

Відкрийте командний рядок і встановіть Node-RED за допомогою цієї команди:

```
npm install -g node-red
```

Тепер запустіть ваш Node-RED за допомогою цієї команди:

```
node-red
```

Для перевірки правильності роботи нам потрібно зайти на сайт <http://localhost:1880>

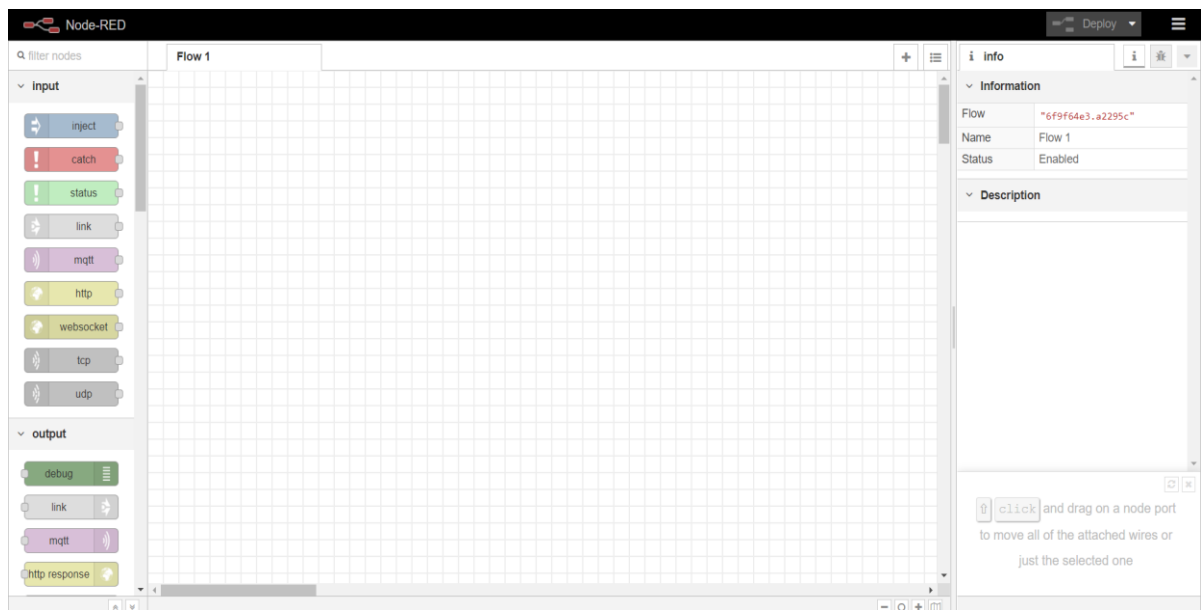


Рисунок 3.4 – Середовище Node-Red

Наступним кроком встановлюємо dashboard і serialport.

Встановити їх можна за допомогою введення в командній строці наступних команд:

- dashboard – `npmi node-red-dashboard`;
- serialport – `npmi node-red-node-serialport`.

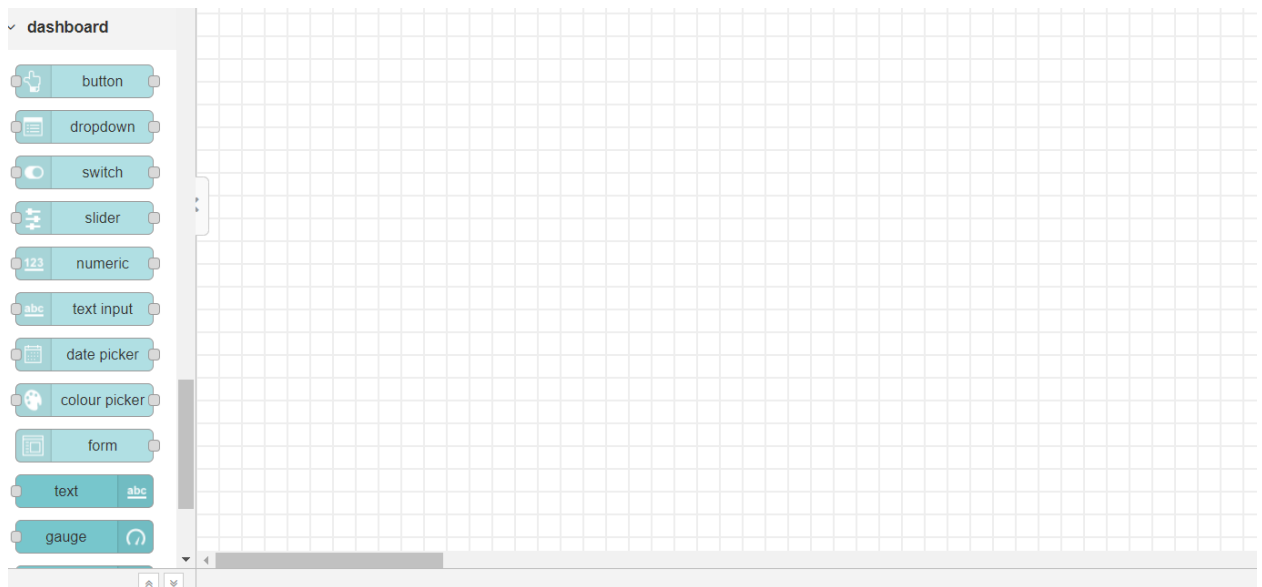


Рисунок 3.5 – Панель dashboard в Node-RED

Збираємо просту модель з світлодіодом на Arduino рис. 3.6.

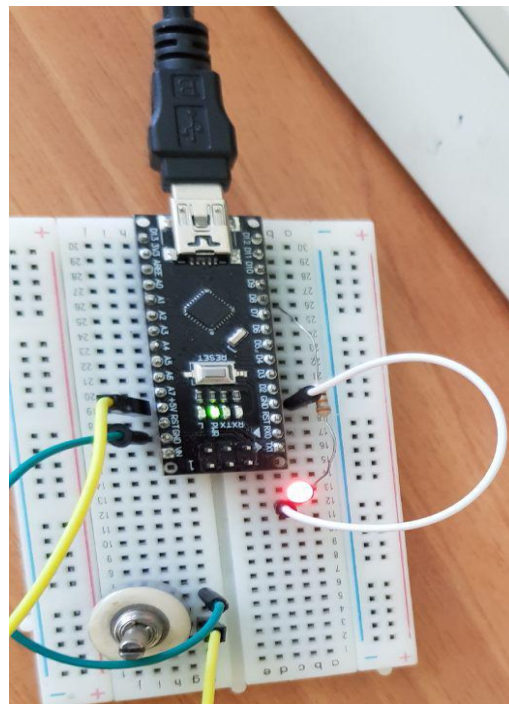


Рисунок 3.6 – Зібрана модель Arduino з світлодіодом

Завантажуємо лістинг коду з наведеного вище в прикладі. Відповідно до даного скетчу, додамо необхідні компоненти з середовища Node-Red, для управління нашою моделлю. До них входять: два порти COM4, один на вхід, а інших на вихід; debug, щоб подивитися як працює дана схема; switch, за

допомогою якого ми надсилаємо строку з числами; дві кнопки 6 та 7, які підключені до switch, завдяки ним ми і будемо надсилати певні сигнали.

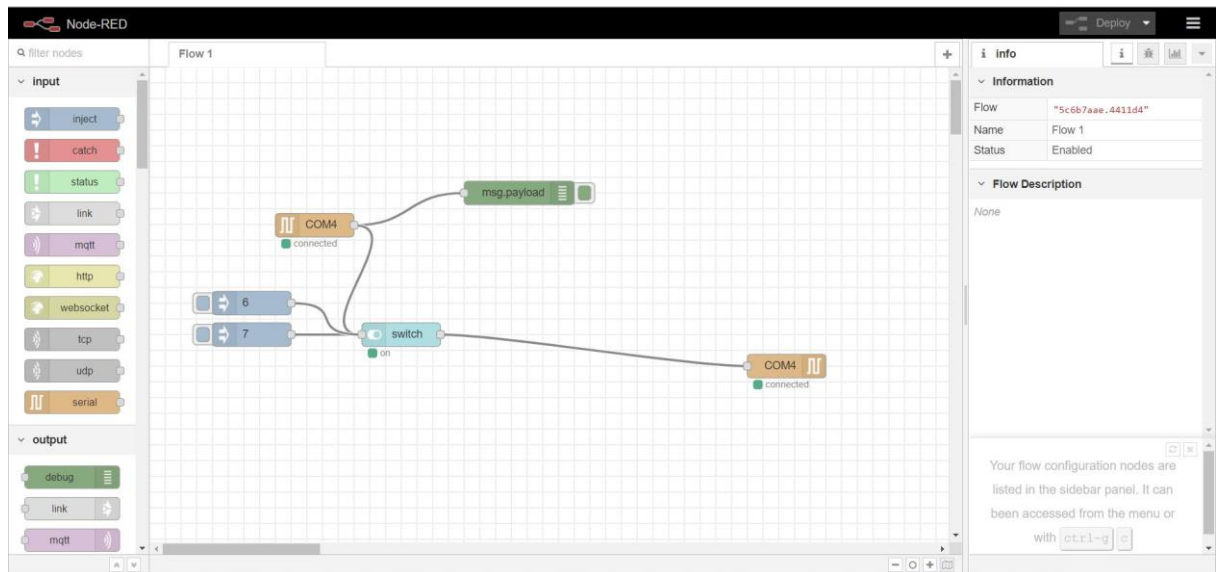


Рисунок 3.7 – Модель для керування Arduino

Налаштуємо елементи COM4, обравши правильний порт і вказати швидкість, таку ж саму як і в коді, в нашому випадку 9600.

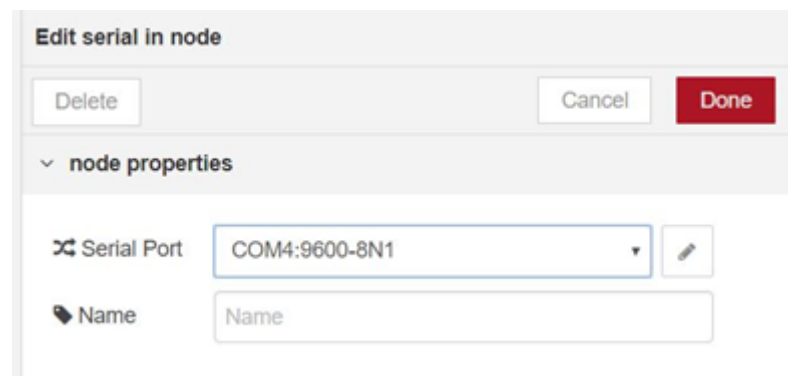


Рисунок 3.8 – Налаштування порта

Для правильного функціонування, необхідно змінити параметри свіча. Нам потрібно задати, щоб він відсилав числову строку.

**Edit switch node**

Delete Cancel Done

▼ **node properties**

Group [Home] Default

Size auto

Label switch

Tooltip optional tooltip

Icon Default

→ If **msg** arrives on input, pass through to output: ☒

☒ When clicked, send:

On Payload ▼ a<sub>z</sub> 6

Off Payload ▼ a<sub>z</sub> 7

Topic

Name

> **node settings**

Рисунок 3.9 – Налаштування Switch

В результаті запуску програми при натисненні кнопки 6, діод має почати світитися, а при натисканні кнопки 7 він перестане світитись.

### Алгоритм роботи програми

1. Мікроконтролер отримує команду від комп'ютера.
2. Виконує аналіз команди.
3. В залежності від команди відсилає символ підтвердження, змінює алгоритм роботи світлода або вимикає його.

### **Контрольні питання**

1. Поясніть призначення програмного середовища Node-RED.
2. В чому полягає особливість встановлення Node-RED?
3. Поясніть за що відповідає команда `npmi node-red-dashboard`?
4. Яку команду необхідно ввести для запуску `serialport`?
5. Для чого потрібно змінювати налаштування `Switch`?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

### 3.3 Лабораторна робота №3. Підключення Wi-Fi модуля ESP8266 до сервера Thinger.io

**Мета:** підключити плату Arduino з інтегрованим модулем ESP8266 WiFi до серверу Thinger.io.

**Завдання:** зібрати схему і написати програму, що буде обмінюватися інформацією з сервером Thinger.io.

**Обладнання:** мікроконтролер Arduino; проводи; макетна плата; USB – кабель; світлодіоди.

#### Загальні відомості

Thinger.io – це платформа з відкритим вихідним кодом для Інтернету речей, підключення та керування продуктами Internet of Things за лічені хвилини. Готова до масштабування інфраструктура для підключення мільйонів пристроїв. Можна керувати пристроями за допомогою простої у використанні адміністративної консолі або інтегрувати їх у бізнес-логіку з REST API [13].

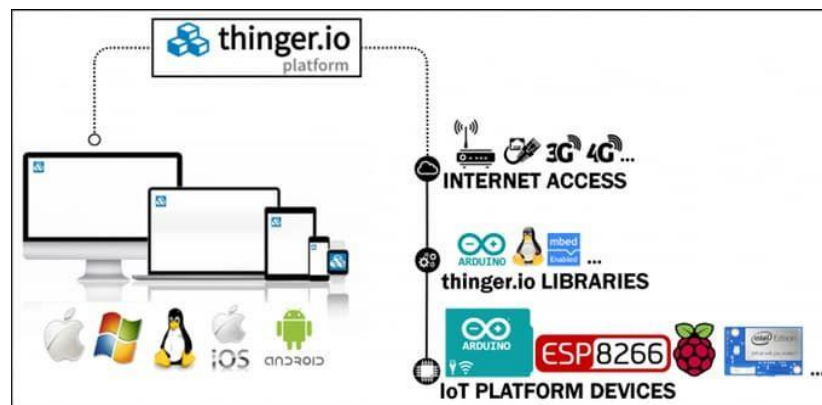


Рисунок 3.10 – Використання платформи Thinger

#### Особливості платформ IoT Thinger

Відкритий вихідний код. Можливість встановити сервер в власній хмарі і використовувати бібліотеки Open Source для підключення пристроїв.



Легке кодування. Ввімкнення світла дистанційно з Інтернету або зчитування значень датчика, яке вимагає однієї лінії коду на мікроконтролері.

Агностик апаратного забезпечення. Підключити можна Arduino, ESP8266, Raspberry Pi, Intel Edison.

Для виробників. Виробники можуть безкоштовно зареєструватися, щоб розпочати створення своїх IoT-проектів за лічені хвилини, використовуючи хмарну інфраструктуру.

Хмарна платформа. Хоча платформа Thinger.io є платформою з відкритим кодом для Інтернету речей, вона забезпечує готову до використання масштабну хмарну інфраструктуру для підключення. Виробники і компанії можуть вести контроль за своїми засобами з Інтернету швидко, не турбуючись про необхідність використання хмарної інфраструктури.

### Приклад коду

```
#include <SPI.h>
#include <ESP8266WiFi.h>
#include <ThingerWifi.h>

#define USERNAME "your_user_name"
#define DEVICE_ID "your_device_id"
#define DEVICE_CREDENTIAL "your_device_credential"

#define SSID "your_wifi_ssid"
#define SSID_PASSWORD "your_wifi_ssid_password"

ThingerWifi thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
    pinMode(BUILTIN_LED, OUTPUT);

    thing.add_wifi(SSID, SSID_PASSWORD);

    // resource input example (i.e. turning on/off a light, a
    relay, configuring a parameter, etc)
    thing["led"] << [](pson& in){ digitalWrite(BUILTIN_LED, in
? HIGH : LOW); };

    // resource output example (i.e. reading a sensor value)
    thing["millis"] >> [](pson& out){ out = millis(); };

    // resource input/output example (i.e. passing input values
    and do some calculations)
    thing["in_out"] = [](pson& in, pson& out){
        out["sum"] = (long)in["value1"] + (long)in["value2"];
```

```

        out["mult"] = (long)in["value1"] * (long)in["value2"];
    };
}

void loop() {
    thing.handle();
}

```

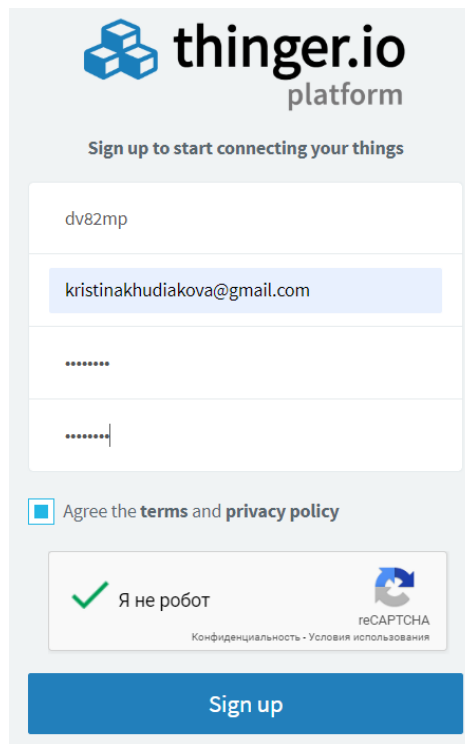
### Хід виконання роботи

1. Зареєструватися на платформі Thingier.io.
2. Підключити бібліотеку.
3. Зібрати макет відповідно до завдання (рис. 3.16).
4. Підключити схему до живлення (3.3 В).
5. Завантажити програму в мікроконтролер Arduino.
6. Перевірити правильність роботи програми.

### Завдання

1. Реєстрація

Для початку потрібно зареєструватися на платформі Thingier.io за посиланням <https://thingier.io> . Приклад показано на рис. 3.11.



The image shows the registration page for Thingier.io. At the top is the logo and the text 'Sign up to start connecting your things'. The form contains the following elements:

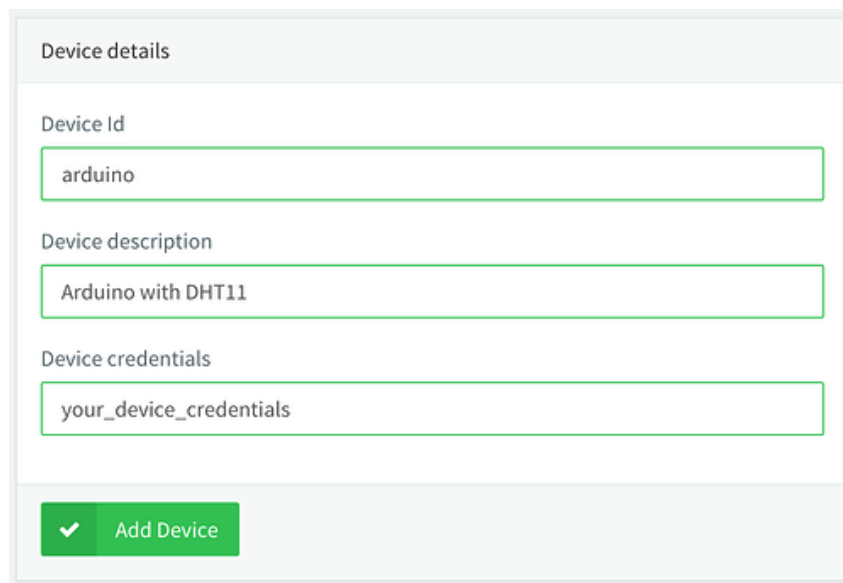
- A text input field with the value 'dv82mp'.
- An email input field with the value 'kristinakhudiakova@gmail.com'.
- A password input field with masked characters '.....'.
- A second password input field with masked characters '.....'.
- A checkbox labeled 'Agree the terms and privacy policy'.
- A reCAPTCHA widget showing a green checkmark, the text 'Я не робот', and the reCAPTCHA logo.
- A blue 'Sign up' button at the bottom.

Рисунок 3.11 – Приклад реєстрації

## 2. Додавання пристрою до платформи Thinger.io

Після входу в "Панель управління консолі" перейти до розділу "Пристрої" (Devices), який з'явиться в меню зліва. У цьому розділі будуть перераховані зареєстровані пристрої та буде відображатися деяка інформація про його підключення.

На цьому екрані натиснути "Додати пристрій" (Add Device), який відкриє форму, що додає деяку інформацію про пристрій.



Device details

Device Id

arduino

Device description

Arduino with DHT11

Device credentials

your\_device\_credentials

✓ Add Device

Рисунок 3.12 – Деталі пристрою

Після успішного додавання пристрою буде видно відповідне повідомлення (рис. 3.13).

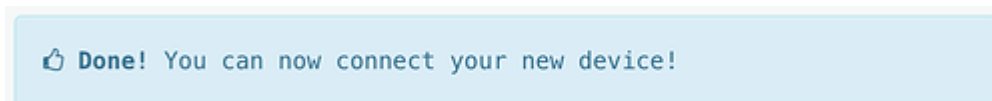


Рисунок 3.13 – Повідомлення про успішне додавання пристрою

Після повернення до списку пристроїв буде видно, що новий пристрій створено (рис. 3.14).

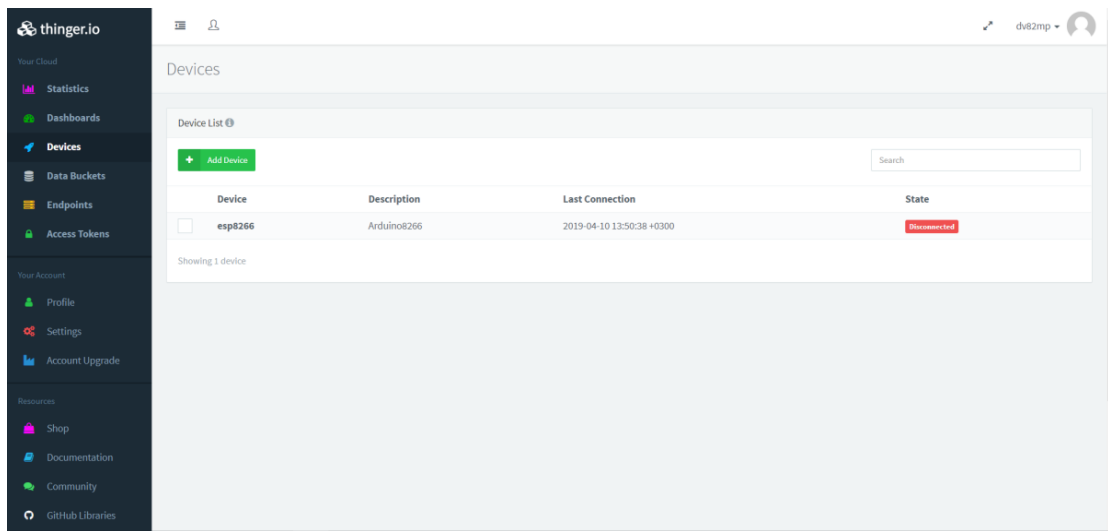


Рисунок 3.14 – Панель управління

Тепер можна використовувати новий ідентифікатор пристрою і облікові дані пристрою для підключення нового пристрою.

### 3. Відкриття панелі моніторингу пристрої

Можна спробувати натиснути на назву пристрою для відкриття його інформаційної панелі, яка наразі відображає певну інформацію, наприклад, стан підключення, час підключення, байти вгору і вниз і стан активності в реальному часі.

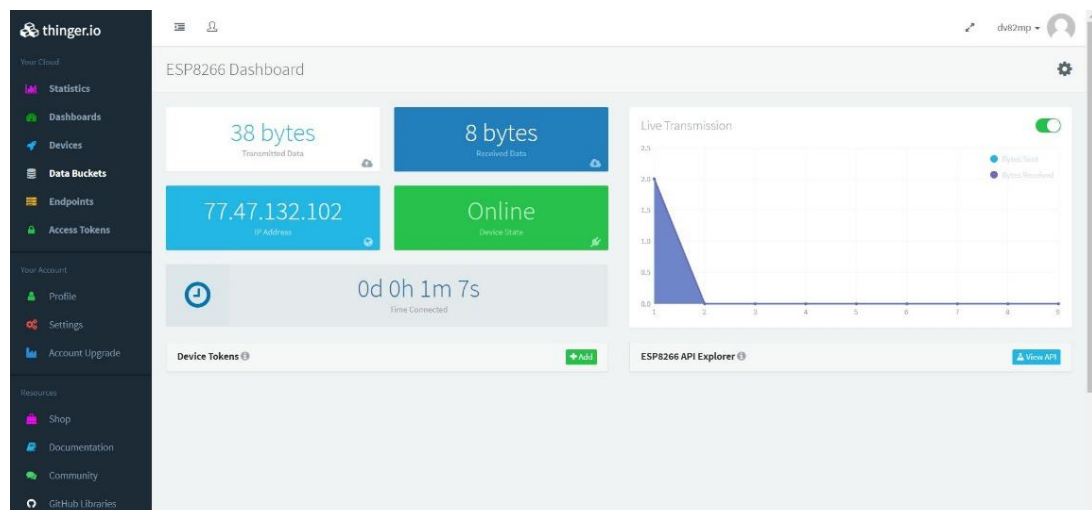


Рисунок 3.15 – Панель моніторингу пристрою

#### 4. Встановлення бібліотеки

За посиланням завантажуюмо бібліотеку <http://docs.thinger.io/arduino/#installation>.

Файл бажано перейменувати з Arduino-Library-master.zip на thinger.zip. Останнім кроком є імпортування zip-бібліотеки за допомогою Arduino IDE. Цей крок розпакує та скопіює бібліотеку zip у папку бібліотек Arduino.

Скетч > Підключити бібліотеку > Додати бібліотеки .ZIP

Тепер бібліотека повинна бути доступна з деякими прикладами за замовчуванням.

#### 5. Запуск прикладного проекту

Тепер IDE готовий до компіляції коду для плат ESP8266 і роботи з платформою. Завантажимо приклад коду для ESP8266, який наведено в прикладі:

```
#include <ThingerESP8266.h>
int pin1 = 2;
int anal = A0;
int an = 0;
#define USERNAME "dv82mp"
#define DEVICE_ID "esp8266"
#define DEVICE_CREDENTIAL "polokoio"
#define SSID "DESKTOP-YURI"
#define SSID_PASSWORD "77777777"
ThingerESP8266 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
    pinMode(pin1, OUTPUT);

    thing.add_wifi(SSID, SSID_PASSWORD);
    // digital pin control example (i.e. turning on/off a light,
    a relay, configuring a parameter, etc)
    thing["led"] << digitalPin(pin1);

    // resource output example (i.e. reading a sensor value)
    thing["millis"] >> outputValue(analogRead(anal));

    // more details at http://docs.thinger.io/arduino/
}
void loop() {
    thing.handle();
}
```

Деякі константи за замовчуванням називаються USERNAME, DEVICE\_ID і DEVICE\_CREDENTIAL, які потрібно заповнити інформацією,

наданою в процесі реєстрації пристрою. Також є конфігурація точки доступу WiFi. Потрібно змінити SSID з потрібним Wifi ім'ям, а SSID\_PASSWORD з паролем Wifi. На цьому етапі програмують пристрій ESP8266 і підключають до платформи (рис. 3.17).

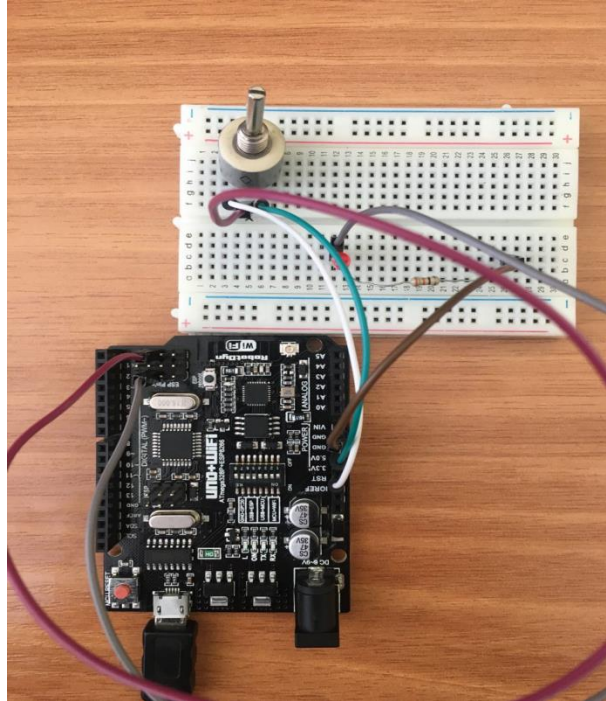


Рисунок 3.16 – Приклад підключення модуля Uno+Wi-Fi у режимі Wi-Fi

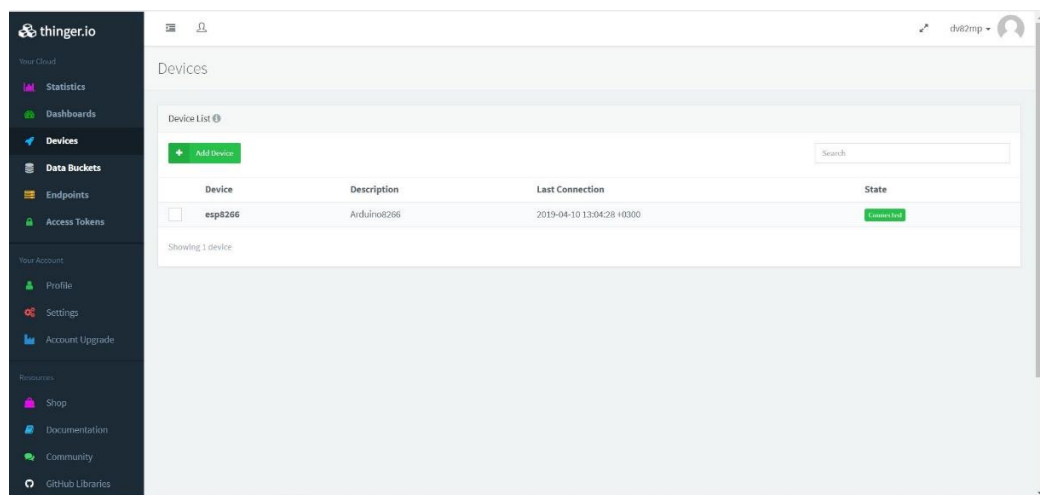


Рисунок 3.17 – Пристрій підключений до платформи

Тепер можна перевірити список своїх пристроїв, ввести в інформаційну панель пристрою, натиснувши на назву пристрою, і отримати доступ до

інформаційної панелі API (натиснувши на кнопку нижче), де можна взаємодіяти з визначеними ресурсами (led, millis, in\_out). У світлодіодному ресурсі можна вмикати і вимикати світлодіод. Це приклад того, як надсилати інформацію на пристрій (рис. 3.18).

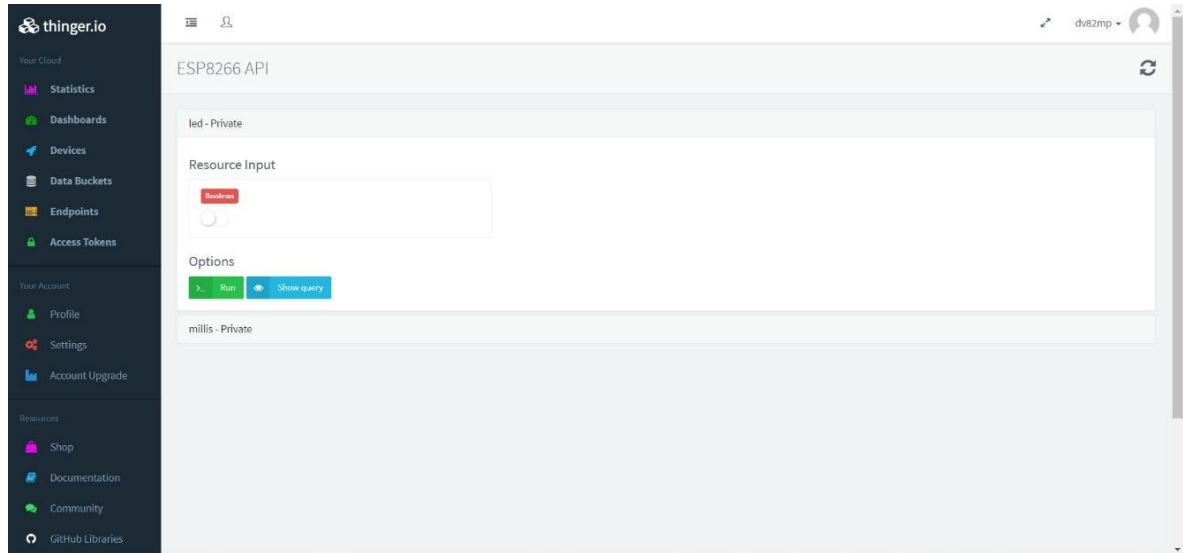


Рисунок 3.18 – Вигляд світлодіодного ресурсу

Millis ресурс є прикладом читання значень з пристрою, який надає деякі дані. У цьому випадку з пристрою зчитуються поточні мілісекунди з початку роботи плати ESP8266.

### Алгоритм роботи програми

1. Мікроконтролер отримує команду від комп'ютера.
2. Виконує аналіз команди.
3. В залежності від команди відсилає символ підтвердження, змінює алгоритм роботи світлода або вимикає його.

### Контрольні питання

1. Поясніть призначення програмного середовища Node-RED.
2. В чому полягає особливість встановлення Node-RED?
3. Поясніть за що відповідає команда `npm i node-red-dashboard`?
4. Яку команду необхідно ввести для запуску serialport?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

### 3.4 Лабораторна робота №4. Підключення модуля ESP8266 до платформи Ubidots через HTTP

**Мета:** підключити модуль ESP8266 Wi-Fi до серверу ubidots.com та надсилати дані в/з Ubidots за допомогою модуля NodeMCU Firmware.

**Завдання:** зібрати схему і написати програму що буде обмінюватися інформацією з обраним сервером ubidots.com.

**Обладнання:** мікроконтролер Arduino; проводи; макетна плата; USB – кабель.

#### Загальні відомості

Зовнішній вигляд мікроконтролеру ESP8266 NodeMCU з модулем Wi-Fi наведено на рис. 3.19.

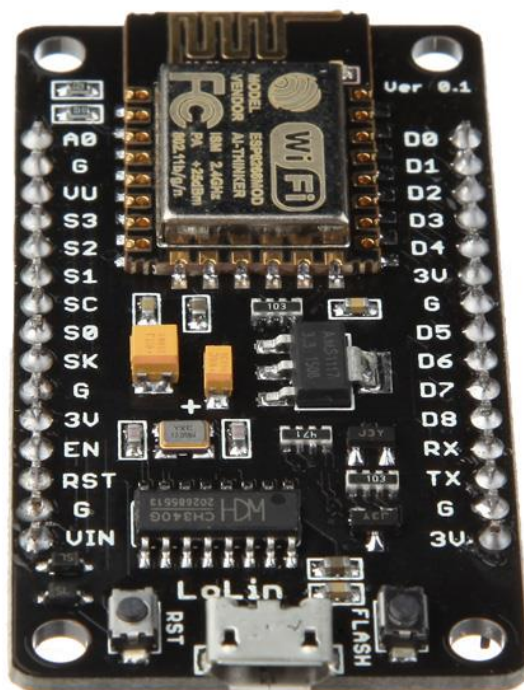


Рисунок 3.19 – Мікроконтролер ESP8266 NodeMCU з модулем Wi-Fi

Плата для розробки, побудована на мікроконтролері ESP8266, об'єднує порти вводу-виводу GPIO, PWM контролер, інтерфейс I2C, 1-провідний інтерфейс та АЦП. В якості програмного забезпечення встановлено NodeMCU Firmware [14].



## **Необхідне програмне забезпечення**

- NodeMCU version 1.0;
- Arduino IDE 1.6.5 або вище;
- Ubidots library;
- Ubidots account або STEM License.

## **Хід виконання роботи**

1. Створити запис Ubidots;
2. Встановити платформу ESP8266 за допомогою Arduino Board Manager;
3. Підключити бібліотеку.
4. Зібрати макет відповідно до завдання.
5. Підключити схему до живлення.
6. Завантажити програму в мікроконтролер Arduino.
7. Перевірити правильність роботи програми.

## **Завдання**

1. Створити Ubidots запис.

Для створення Ubidots запису для особистого або освітнього вжитку необхідно перейти за посиланням:

[https://industrial.ubidots.com/accounts/signup\\_industrial/](https://industrial.ubidots.com/accounts/signup_industrial/).

2. Налаштування Arduino IDE.

Щоб мати можливість працювати з платформою NodeMCU ESP8266 в Arduino IDE, потрібно встановити платформу ESP8266 за допомогою Arduino Board Manager.

За допомогою встановленої платформи ESP8266 необхідно вибрати пристрій ESP8266, який використано у роботі.

У даному випадку це “NodeMCU 1.0 (модуль ESP12E)”. Щоб вибрати вказану плату з Arduino IDE, потрібно зайти Інструменти> Плата “NodeMCU 1.0 (модуль ESP12E)”.

Далі необхідно завантажити та встановити бібліотеку Ubidots за посиланням: <https://github.com/ubidots/ubidots-esp8266/> [14].

Детальне пояснення щодо встановлення бібліотек за допомогою Arduino IDE див. у посібнику за посиланням: <http://help.ubidots.com/technical-resources/setting-up-the-arduino-ide-for-ubidots>.

### 3. Надсилання даних (POST) на Ubidots.

Наступним прикладом можна імітувати випадкові показання, взяті з NodeMCUESP8266 на Ubidots.

Для початку публікації значень у Ubidots необхідно відкрити ID Arduino і вставити зразок коду нижче.

Після вставки коду обов'язково потрібно призначити такі параметри:

- SSID (ім'я WiFi) та пароль доступного мережевого з'єднання.
- Ubidots TOKEN, який було отримано при реєстрації.

```
/*
*****

* Include Libraries
*****/

#include "Ubidots.h"

/*
*****

* Define Instances and Constants
*****/

const char* UBIDOTS_TOKEN = "..."; // Put here your Ubidots
TOKEN
const char* WIFI_SSID = "..."; // Put here your Wi-Fi SSID
const char* WIFI_PASS = "..."; // Put here your Wi-Fi password

Ubidots ubidots(UBIDOTS_TOKEN, UBI_HTTP);

/*
*****

* Main Functions
*****/

void setup() {
    Serial.begin(115200);
    ubidots.wifiConnect(WIFI_SSID, WIFI_PASS);
}
```

```

    // ubidots.setDebug(true);    // Uncomment this line for
    printing debug messages
}

void loop() {
    float value1 = random(0, 9) * 10;
    float value2 = random(0, 9) * 100;
    float value3 = random(0, 9) * 1000;
    ubidots.add("Variable_Name_One", value1);    // Change for
    your variable name
    ubidots.add("Variable_Name_Two", value2);
    ubidots.add("Variable_Name_Three", value3);
    bool bufferSent = false;
    bufferSent = ubidots.send();    // Will send data to a device
    label that matches the device Id
    if (bufferSent) {
        // Do something if values were sent properly
        Serial.println("Values sent by the device");
    }
    delay(5000);
}

```

Після перевірки код завантажується у Arduino UNO + Ethernet Shield.

Щоб перевірити зв'язок пристрою та надісланих даних, необхідно відкрити послідовний монітор, щоб побачити журнали підключення.

Далі необхідно підтвердити свої дані в Ubidots. Тепер має бути видно, що дані розміщені у обліковому записі Ubidots, розташованому на пристрої під назвою "arduino-ethernet" (рис. 3.20).

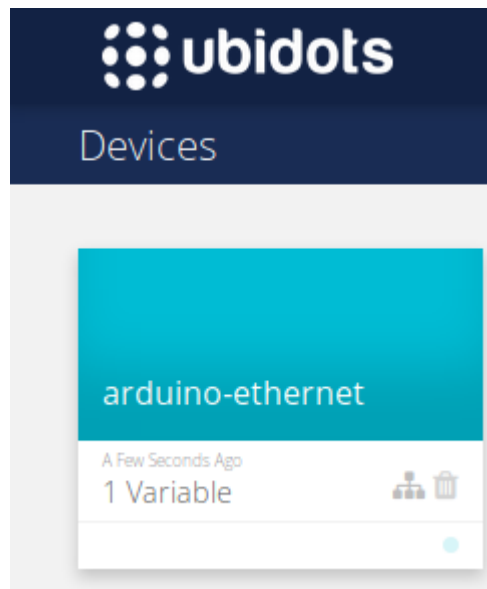


Рисунок 3.20 – Дані розміщені у обліковому записі Ubidots на пристрої під назвою "arduino-ethernet"

#### 4. Отримання (GET) даних від Ubidots.

За допомогою наступного зразкового коду можна отримати значення від Ubidots, щоб почати контролювати будь-який актив, який ви хочете [15].

Щоб почати отримувати значення з Ubidots, необхідно відкрити Arduino IDE і вставити зразок коду нижче. Після вставки коду обов'язково необхідно призначити такі параметри:

Ubidots TOKEN

Мітка пристрою, яка містить змінну, яку потрібно отримати.

Мітка змінної, яку потрібно отримати.

```

/*****
 * Libraries included
 *****/

#include <Ethernet.h>
#include <SPI.h>
#include <UbidotsEthernet.h>

/*****
 * Constants and objects

```

```

*****/

/* Assigns the Ubidots parameters */
char const * TOKEN = "Assign_your_Ubidots_TOKEN_here"; //
Assign your Ubidots TOKEN

char const * DEVICE_LABEL = "Assign_device_label_here"; //
Assign the unique device label

char const * VARIABLE_LABEL = "Assign_variable_label_here";
// Assign the unique variable label to get the last value


/* Enter a MAC address for your controller below */
/* Newer Ethernet shields have a MAC address printed on a
sticker on the shield */
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };


/* initialize the instance */
Ubidots client(TOKEN);


/*****
 * Main Functions
 *****/

void setup() {
  Serial.begin(9600);
  //client.setDebug(true); // uncomment this line to visualize
the debug message
  /* start the Ethernet connection */
  Serial.print(F("Starting ethernet..."));
  if (!Ethernet.begin(mac)) {
    Serial.println(F("failed"));
  } else {
    Serial.println(Ethernet.localIP());
  }
  /* Give the Ethernet shield a second to initialize */
  delay(2000);
  Serial.println(F("Ready"));
}

void loop() {
  Ethernet.maintain();
}

```

```

/* Getting the last value from a variable */
float      value      =      client.getValue (DEVICE_LABEL,
VARIABLE_LABEL);
/* Print the value obtained */
Serial.print("the value received is:  ");
Serial.println(value);
delay(5000);
}

```

Після перевірки потрібно завантажити код на плату, виконуючи ті ж самі кроки, що наведені вище в кроці POST.

Щоб перевірити підключення пристрою та отримані дані, необхідно відкрити послідовний монітор. На послідовному моніторі можна побачити останнє значення, отримане в Ubidots змінної (рис. 3.21).

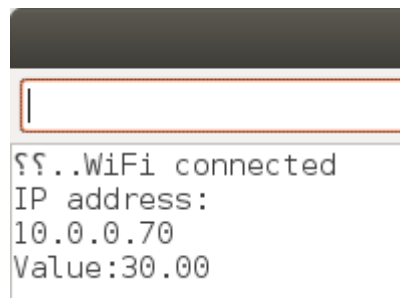


Рисунок 3.21 – Перевірка отриманих значень на послідовному моніторі

### Контрольні питання

1. Поясніть призначення платформи Ubidots.
2. В чому полягає особливість роботи з платформою NodeMCU ESP8266 в Arduino IDE?
3. Як відбувається надсилання даних на платформу Ubidots?
4. Що необхідно виконати для отримання даних від платформи Ubidots?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

### 3.5 Лабораторна робота №5. Реалізація протоколу MQTT за допомогою Arduino

**Мета роботи:** дослідження можливостей протоколу MQTT для передавання даних.

**Завдання:** реалізувати протокол MQTT за допомогою Arduino.

**Обладнання:** плата Arduino Uno R3; плата Arduino Ethernet shield (W5100); проводи; світлодіод з резистором (330 Ом); USB-кабель; датчик температури та вологості DHT11.

#### Загальні відомості

Відповідно до специфікації MQTT 3.1.1: "MQTT – це простий протокол обміну повідомленнями через брокера за схемою публікації/підписки, головна мета протоколу забезпечити відкритість, простоту, мінімальні вимоги до ресурсів і зручність застосування".

Перераховані переваги дозволяють застосовувати протокол MQTT в системах M2M (Machine-to-Machine) та IIoT (Industrial Internet of Things).

Протокол MQTT (MQ Telemetry Transmission) працює над TCP/IP (та/або іншими мережевими протоколами, орієнтованими на з'єднання) для передавання повідомлень між взаємопов'язаними пристроями.

MQTT повідомлення складається з декількох частин:

- фіксований заголовок (у всіх повідомленнях);
- змінний заголовок (присутній тільки в певних повідомленнях);
- дані, "навантаження" (присутні тільки в певних повідомленнях)

Пристрій (вузол) може публікувати повідомлення та/або підписуватися на повідомлення від брокера (сервера) MQTT. Брокер MQTT отримує опубліковані повідомлення, слідкує за підписниками та надсилає повідомлення на потрібний пристрій. Принцип роботи брокера протоколу MQTT наведено на рис. 3.22.

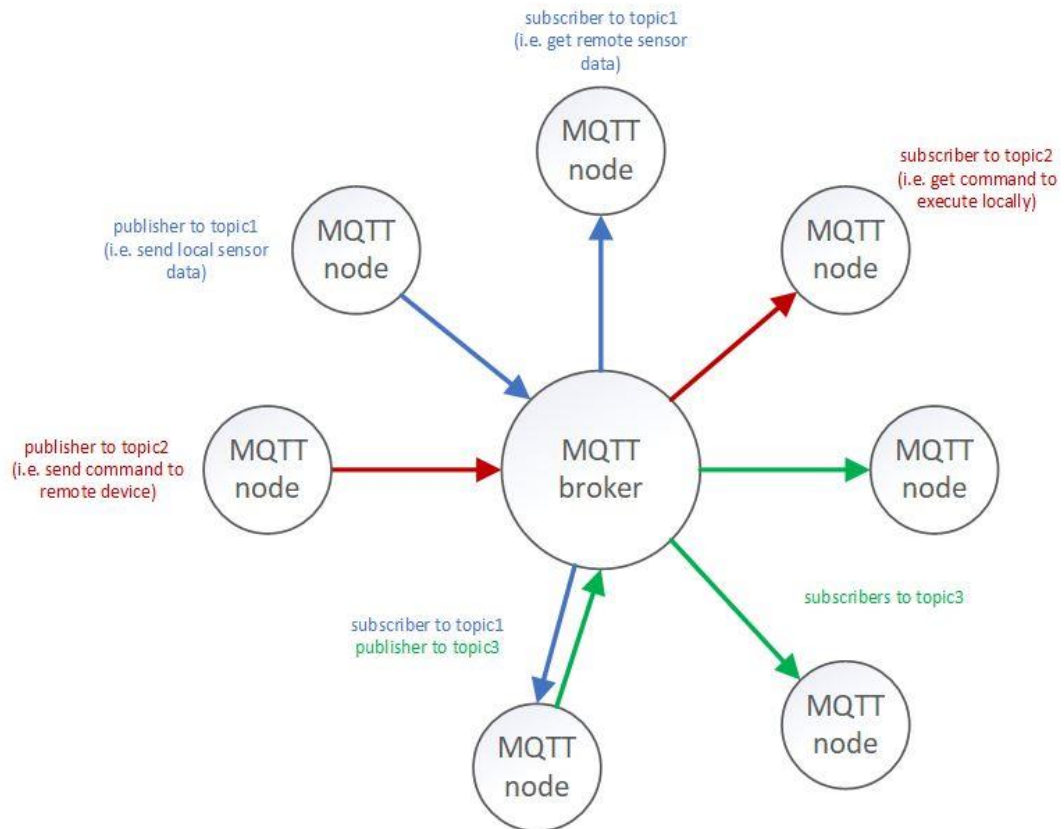


Рисунок 3.22 – Принцип роботи брокера протоколу MQTT

Доступно багато MQTT-брокерів, як хмарних (тобто на AWS, Azure, ThingSpeak тощо), так і самостійно розміщених. Для цієї лабораторної роботи ми будемо використовувати рішення з відкритим кодом, що розміщено самостійно: сервер Mosquitto. Цей повністю сумісний сервер MQTT доступний як для платформ Linux, так і для Windows, а також утиліти командного рядка для публікації та підписки [15].

#### Хід виконання роботи

1. Зібрати макет відповідно завдання (рис. 3.24).
2. Підключити схему до живлення(5В).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи програми.



## Завдання

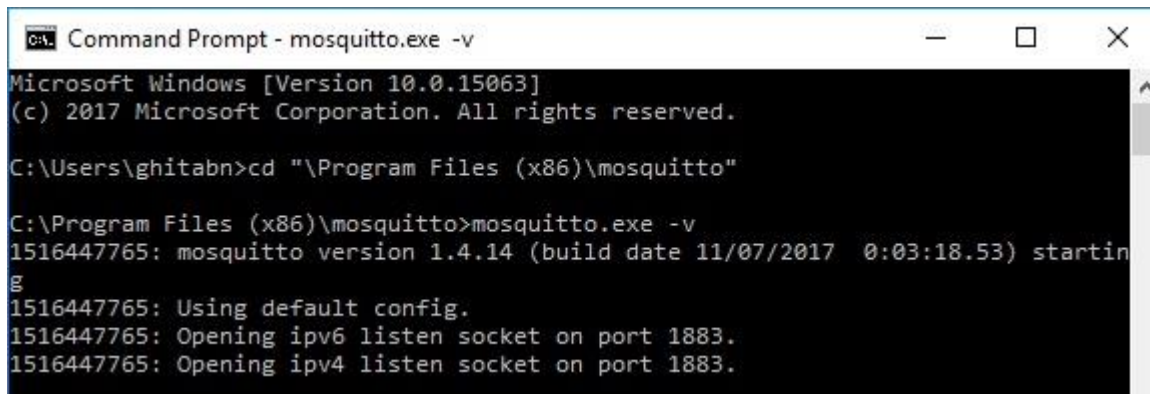
Реалізувати протокол MQTT за допомогою Arduino. Для цієї демонстрації ми будемо використовувати сервер Mosquitto, встановлений на операційну систему Windows 10 (бінарні файли та процедури встановлення доступні на <https://mosquitto.org/>).

За замовчуванням сервер Mosquitto працює на порту 1883 без захисту – це не проблема для цієї демонстрації. Більш детальну інформацію щодо забезпечення каналу передавання можна знайти на веб-сторінці документації на сервер Mosquitto.

Обмежені ресурси (процесор і пам'ять), наявні на платі Arduino Uno, не можуть успішно підтримувати такі реалізації.

У Windows сервер можна запустити автоматично (як сервіс) або вручну з командного рядка. Ручний запуск у багатослівному режимі пропонує більше розуміння того, що відбувається на стороні сервера:

```
cd "\\Program Files (x86)\\mosquitto"
mosquitto.exe -v
```



```
Command Prompt - mosquitto.exe -v
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\ghitabn>cd "\\Program Files (x86)\\mosquitto"

C:\Program Files (x86)\\mosquitto>mosquitto.exe -v
1516447765: mosquitto version 1.4.14 (build date 11/07/2017 0:03:18.53) starting
1516447765: Using default config.
1516447765: Opening ipv6 listen socket on port 1883.
1516447765: Opening ipv4 listen socket on port 1883.
```

Рисунок 3.23 – Вікно командного рядка для запуску серверу вручну

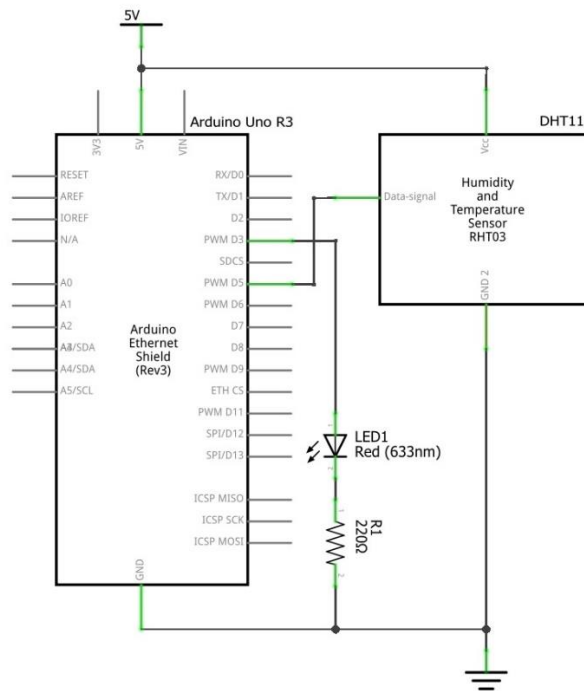


Рисунок 3.24 – Схема підключення Arduino MQTT node

Зовнішній вигляд зібраного макету Arduino MQTT node наведено на рис. 3.25.

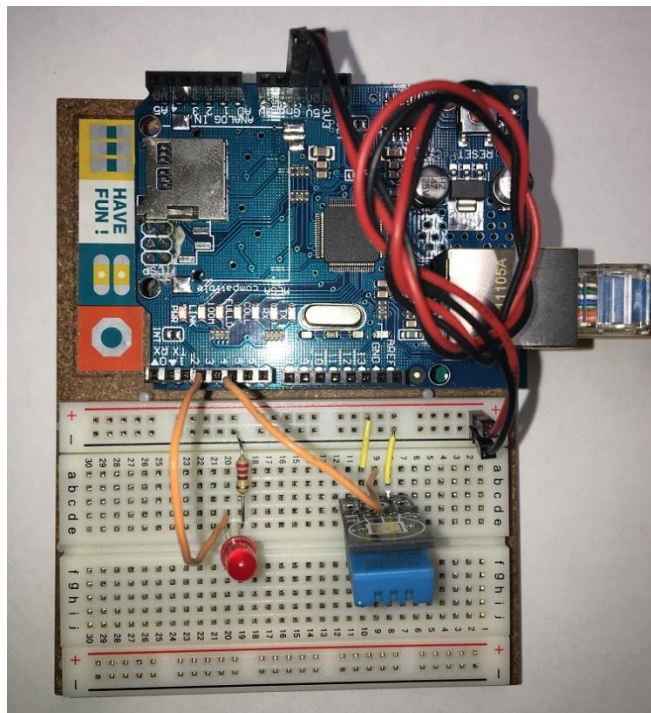


Рисунок 3.25 – Зовнішній вигляд макету Arduino MQTT node

### Завантажуємо код програми Arduino MQTT node.

```
#include <SPI.h>

#include <Ethernet.h>

#include <PubSubClient.h>

#include <DHT.h>


#define ARDUINO_CLIENT_ID "arduino_1" //
Client ID for Arduino pub/sub

#define PUB_TEMP "arduino_1/sensor/temperature_celsius" //
MTTQ topic for temperature [C]

#define PUB_HUMID "arduino_1/sensor/humidity" //
MTTQ topic for humidity

#define SUB_LED "arduino_1/led" //
MTTQ topic for LED

#define PUBLISH_DELAY 3000 //
Publishing delay [ms]

// Hardware setup details

const int ledPin = 3;

const int sensorPin = 5;

const int sensorType = DHT11;

// Networking details

byte mac[] = { 0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 }; //
Ethernet shield (W5100) MAC address

IPAddress ip(192, 168, 2, 105); //
Ethernet shield (W5100) IP address

IPAddress server(192, 168, 2, 114); //
MTTQ server IP address

DHT dht(sensorPin, sensorType);

EthernetClient ethClient;

PubSubClient client(ethClient);

long previousMillis;

void setup()

{
```

```

Serial.begin(9600);

// LED off

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW);

// MTTQ parameters

client.setServer(server, 1883);

client.setCallback(callback);

// Ethernet shield configuration

Ethernet.begin(mac, ip);

delay(1500); // Allow hardware to stabilize

previousMillis = millis();
}

void loop()
{
    if (!client.connected())
        reconnect();

    if (millis() - previousMillis > PUBLISH_DELAY)
    {
        previousMillis = millis();

        float humidity = dht.readHumidity(); // humidity
        float tempC = dht.readTemperature(); // temperature [C]
        char tmpBuffer[20];

        // check if any reads failed and exit early (to try again).
        if (isnan(humidity) || isnan(tempC))
        {
            Serial.println("error reading sensor data");

            return;
        }
    }
}

```

```

else
{
    Serial.print("[sensor data] temperature[C]: ");
    Serial.print(tempC);
    Serial.print(", humidity: ");
    Serial.println(humidity);
    client.publish(PUB_TEMP, dtostrf(tempC, 6, 2, tmpBuffer));
    client.publish(PUB_HUMID, dtostrf(humidity, 6, 2,
tmpBuffer));
}
}
client.loop();
}
void reconnect()
{
    // Loop until reconnected
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection ... ");
        // Attempt to connect
        if (client.connect(ARDUINO_CLIENT_ID)) {
            Serial.println("connected");
            // (re)subscribe
            client.subscribe(SUB_LED);
        } else {
            Serial.print("Connection failed, state: ");
            Serial.print(client.state());
            Serial.println(", retrying in 5 seconds");
            delay(5000); // Wait 5 seconds before retrying
        }
    }
}

```

```

}
// sub callback function
void callback(char* topic, byte* payload, unsigned int length)
{
    Serial.print("[sub: ");
    Serial.print(topic);
    Serial.print("] ");
    char message[length + 1] = "";
    for (int i = 0; i < length; i++)
        message[i] = (char)payload[i];
    message[length] = '\0';
    Serial.println(message);
    // SUB_LED topic section
    if (strcmp(topic, SUB_LED) == 0)
    {
        if (strcmp(message, "on") == 0)
            digitalWrite(ledPin, HIGH);
        if (strcmp(message, "off") == 0)
            digitalWrite(ledPin, LOW);
    }
}

```

Після програмування плати встановлюється зв'язок MQTT клієнта з сервером Mosquitto.

Вікно встановлення зв'язку MQTT клієнта з сервером Mosquitto наведено на рис. 3.26.

```
Nodejs command prompt - mosquitto.exe -v
C:\Users\Yuri>cd "\Program Files\mosquitto"
C:\Program Files\mosquitto>mosquitto.exe -v
1585582384: mosquitto version 1.6.9 starting
1585582384: Using default config.
1585582384: Opening ipv6 listen socket on port 1883.
1585582384: Opening ipv4 listen socket on port 1883.
1585582387: New connection from 192.168.0.110 on port 1883.
1585582387: New client connected from 192.168.0.110 as arduino_1 (p2, c1, k15).
1585582387: No will message specified.
1585582387: Sending CONNACK to arduino_1 (0, 0)
1585582387: Received SUBSCRIBE from arduino_1
          arduino_1/led (QoS 0)
1585582387: arduino_1 0 arduino_1/led
1585582387: Sending SUBACK to arduino_1
1585582387: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/temperature_celsius', ... (6 bytes))
1585582387: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/humidity', ... (6 bytes))
1585582390: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/temperature_celsius', ... (6 bytes))
1585582393: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/humidity', ... (6 bytes))
1585582396: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/temperature_celsius', ... (6 bytes))
1585582399: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/humidity', ... (6 bytes))
1585582399: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/temperature_celsius', ... (6 bytes))
1585582402: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/humidity', ... (6 bytes))
1585582402: Received PUBLISH from arduino_1 (d0, q0, r0, m0, 'arduino_1/sensor/temperature_celsius', ... (6 bytes))
1585582402: Received PINGREQ from arduino_1
1585582402: Sending PINGRESP to arduino_1
```

Рисунок 3.26 – Встановлення зв'язку MQTT клієнта з сервером Mosquitto

Вивести на екран дані температури та вологості с сервера Mosquitto можливо у окремій вікна, використовуючи командні рядки.

Для температури: `mosquitto_sub.exe -h 192.168.0.106 -i cmd_sub -t arduino_1/sensor/temperature_celsius` (рис. 3.27)

Значення змінних:

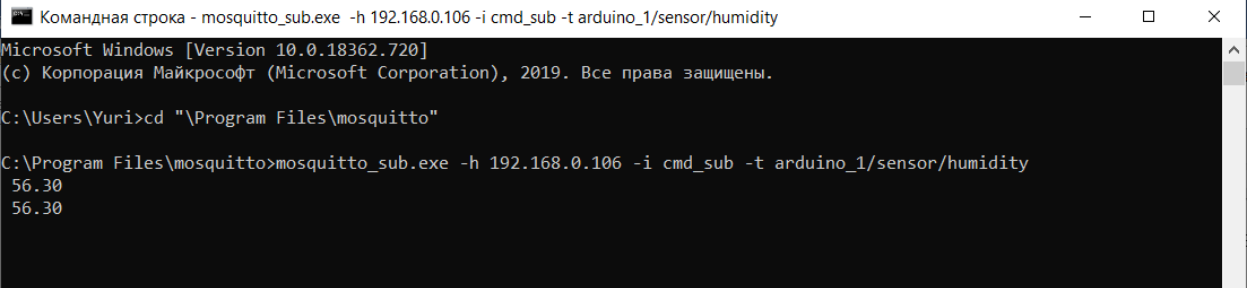
- `-h` (прапор хоста) вказує на сервер, на якому працює Mosquitto;
- `-i` (прапор посвідчення особи) ідентифікує клієнта; необов'язково, якщо відсутній Mosquitto автоматично генерує його;
- `-t` (прапор теми) вказує назву теми;
- `-m` (прапор повідомлення) вказує на повідомлення, яке підлягає публікації.

```
Командная строка - mosquitto_sub.exe -h 192.168.0.106 -i cmd_sub -t arduino_1/sensor/temperature_celsius
Microsoft Windows [Version 10.0.18362.720]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Yuri>cd "\Program Files\mosquitto"
C:\Program Files\mosquitto>mosquitto_sub.exe -h 192.168.0.106 -i cmd_sub -t arduino_1/sensor/temperature_celsius
22.50
22.50
22.50
```

Рисунок 3.27 – Вікно командного рядка для виведення даних температури

Для вологості: `mosquitto_sub.exe -h 192.168.0.106 -i cmd_sub -t arduino_1/sensor/humidity` (рис. 3.28).



```
Командная строка - mosquitto_sub.exe -h 192.168.0.106 -i cmd_sub -t arduino_1/sensor/humidity
Microsoft Windows [Version 10.0.18362.720]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Yuri>cd "\Program Files\mosquitto"

C:\Program Files\mosquitto>mosquitto_sub.exe -h 192.168.0.106 -i cmd_sub -t arduino_1/sensor/humidity
56.30
56.30
```

Рисунок 3.28 – Вікно командного рядка для виведення даних вологості

### Алгоритм роботи програми

1. Мікроконтролер отримує команду від комп'ютера.
2. Виконує аналіз команди.
3. В залежності від команди відсилає символ підтвердження, змінює алгоритм роботи світлода або вимикає його.

### Контрольні питання

1. Поясніть алгоритм роботи протоколу MQTT.
2. Опишіть принцип роботи брокера протоколу MQTT.
3. Поясніть структуру повідомлення протоколу MQTT.
4. Які є варіанти запуску серверу Mosquitto для протоколу MQTT?
5. Яку команду необхідно ввести для запуску серверу Mosquitto?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)



### 3.6 Лабораторна робота №6. Bluetooth модуль ESP32

**Мета:** дослідження роботи Bluetooth модуля ESP32.

**Завдання:** підключити Bluetooth модуль ESP32 до ПК та перевірити роботу модуля програмним додатком для смартфона BLE Scanner.

**Обладнання:** мікроконтролер Arduino; проводи; модуль ESP32; макетна плата; USB – кабель.

#### Загальні відомості

Безпроводова технологія Bluetooth з низьким енергоспоживанням (англ. Bluetooth Low Energy, Bluetooth LE, BLE, представлена також як Bluetooth Smart) – випущена в грудні 2015 року версія специфікації ядра безпроводової технології Bluetooth, найбільш істотною перевагою якої є низьке енергоспоживання, середнє енергоспоживання та енергоспоживання в режимі сну.

Пристрої, що використовують Bluetooth з низьким енергоспоживанням, будуть споживати менше енергії, ніж інші Bluetooth-пристрої попередніх поколінь. У багатьох випадках пристрої зможуть працювати більше року на одній мініатюрної батареї типу таблетка без підзарядки. Таким чином, можна буде мати, наприклад, невеликі датчики, що працюють безперервно (наприклад датчик температури), спілкуються з іншими пристроями, такими як стільниковий телефон або КПК.

Нова версія специфікації Bluetooth дає можливість підтримки широкого діапазону додатків і зменшує розмір кінцевого пристрою для зручного використання в галузях охорони здоров'я, фізкультури і спорту, охоронних систем і домашніх розваг.

Споживаючи менше енергії, технологія Bluetooth з низьким енергоспоживанням пропонує тривале забезпечення зв'язку і з'єднує невеликі пристрої типу датчиків і мобільні пристрої в межах персональних мереж (PAN).

Специфікація Bluetooth 4.0 (і більш пізні) фактично визначає дві безпроводові технології: BR/EDR (класичний Bluetooth, що розвивається, починаючи з першої версії стандарту) і BLE (Bluetooth Low Energy).

Пристрої, в яких застосовано BLE, можуть бути як дворежимні BR / EDR/BLE (називаються Bluetooth Smart Ready), сумісні з класичними Bluetooth-пристроями, так і однорежимні BLE (Bluetooth Smart).

Основними блоками Bluetooth-пристроїв є:

- додаток (англ. application) – реалізує логіку роботи для кінцевого користувача;
- провідний пристрій, хост (англ. host) – надає верхні рівні стека протоколів Bluetooth;
- контролер (англ. controller) – відповідає за нижні рівні Bluetooth.

Комерційні продукти зазвичай використовують одне з наступних апаратних рішень:

- SoC – однокристальна система, що поєднує в собі додаток, хост і контролер, та застосовується в компактних недорогих пристроях, таких як датчики;
- рішення на двох мікросхемах, при якому додаток і хост з'єднані з контролером за допомогою UART, USB, SDIO і т.п. по протоколу HCI, така конфігурація може використовуватися, наприклад, в мобільних пристроях;
- рішення на двох мікросхемах, в якому додаток з'єднується з пристроєм зв'язку (хост і контролер) по пропрієтарного протоколу.

Bluetooth з низьким енергоспоживанням є розширенням специфікації базової безпроводової технології Bluetooth, яка додасть нові функціональні можливості і дозволить створювати додатки для віддаленого управління, медичного спостереження, спортивних датчиків і інших пристроїв.

Bluetooth з низьким енергоспоживанням надає можливість поліпшити існуючі варіанти застосування і функціональні можливості технології Bluetooth.

Відповідні чіпи можуть бути інтегровані в такі продукти, як наручний годинник, безпроводові клавіатури, геймпади і датчики тіла, які можуть підключатися до приймаючих (хостів) пристроїв, таким як мобільні телефони, персональні цифрові помічники (КПК) і персональні комп'ютери (ПК).

На ринку патентованих рішень для забезпечення зв'язку, технологія Bluetooth з низьким енергоспоживанням визначає себе як:

- широко поширений промисловий стандарт протоколів (Bluetooth SIG);
- міжнародно прийнятий промисловий стандарт для передавання даних (IEEE 802.15.1);
- низька ціна завдяки інтеграції мікросхеми;
- сумісність з вже існуючими Bluetooth-пристроями завдяки оновленням.

### **Хід виконання роботи**

1. Налаштувати програмне середовище для роботи;
2. Підключити модуль ESP32 (рис. 3.29) до комп'ютера ;
3. Обрати скетч коду із доступних прикладів у програмному середовищі Arduino.
4. Перевірити правильність роботи програми.

### **Завдання**

Першим кроком для виконання роботи, переходимо на сторінку: <https://github.com/espressif/arduino-esp32>.

На даному сайті, шукаємо «Installation Instructions» і переходимо по першому посиланню ([https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards\\_manager.md](https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md)) .

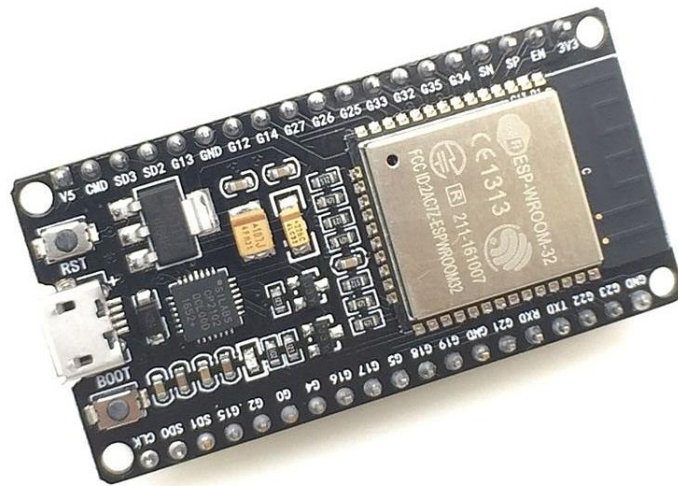


Рисунок 3.29 – Модуль ESP32

Після того, як потрапили на дану сторінку, слідуючи пунктам, налаштовуємо програмне середовище для виконання роботи. Підключаємо плату до комп'ютера. У вкладці інструменти наводимо мишку на поле *Плата-> Менеджер плат -> Dev Module ESP32* (рис. 3.29)

Наступним кроком, обираємо скетч коду із доступним нам у програмному середовищі Arduino (рис. 3.30).

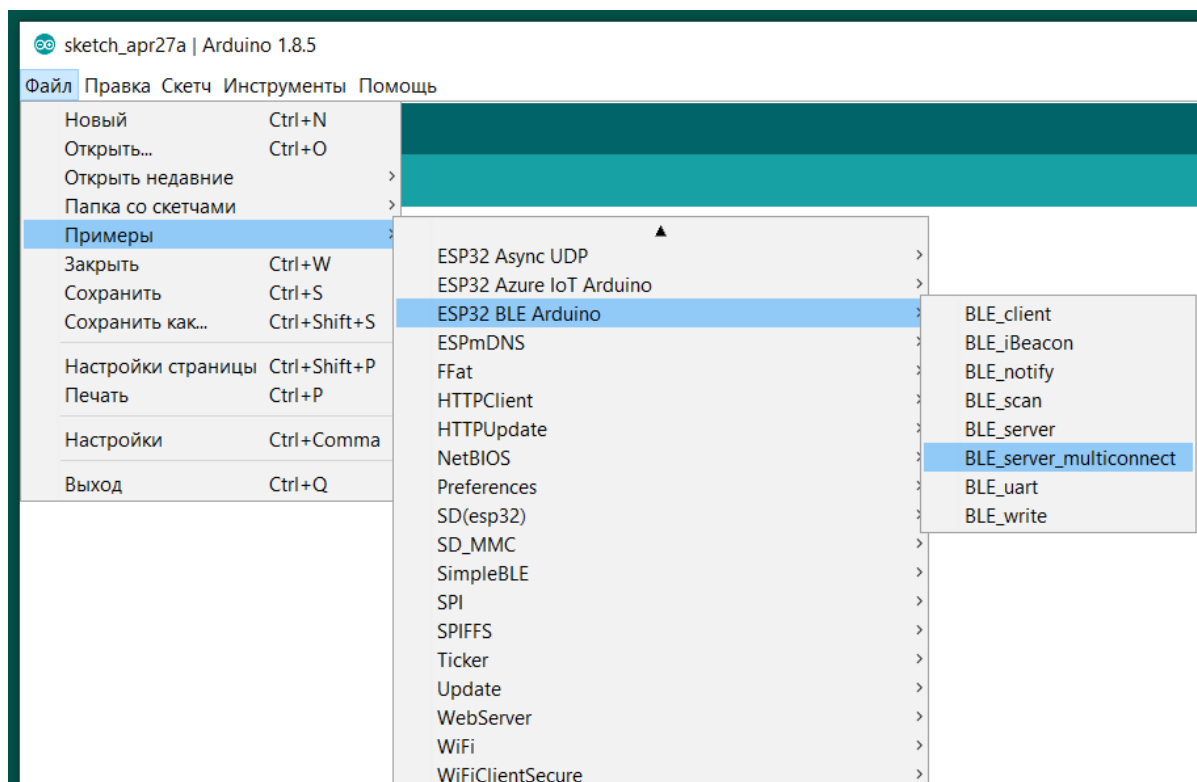


Рисунок 3.30 – Вибір скетчу

Нижче наведено приклад коду, який відкриється у наступному вікні після вибору скетчу, (рис. 3.31):

```
/*
https://github.com/nkolban/esp32-
snippets/blob/master/cpp_utils/tests/BLE%20Tests/SampleNotify.cpp
    Ported to Arduino ESP32 by Evandro Copercini
    updated by chegewara

    Create a BLE server that, once we receive a connection, will send
    periodic notifications.

    The service advertises itself as: 4fafc201-1fb5-459e-8fcc-
    c5c9c331914b

    And has a characteristic of: beb5483e-36e1-4688-b7f5-ea07361b26a8

    The design of creating the BLE server is:
    1. Create a BLE Server
    2. Create a BLE Service
    3. Create a BLE Characteristic on the Service
    4. Create a BLE Descriptor on the characteristic
    5. Start the service.
    6. Start advertising.

    A connect handler associated with the server starts a background
    task that performs notification
    every couple of seconds.
*/
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic = NULL;
bool deviceConnected = false;
bool oldDeviceConnected = false;
uint32_t value = 0;
```

```

// See the following for generating UUIDs:
// https://www.uuidgenerator.net/

#define SERVICE_UUID          "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID  "beb5483e-36e1-4688-b7f5-ea07361b26a8"

class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
        deviceConnected = true;
        BLEDevice::startAdvertising();
    };

    void onDisconnect(BLEServer* pServer) {
        deviceConnected = false;
    }
};

void setup() {
    Serial.begin(115200);

    // Create the BLE Device
    BLEDevice::init("ESP32");

    // Create the BLE Server
    pServer = BLEDevice::createServer();
    pServer->setCallbacks(new MyServerCallbacks());

    // Create the BLE Service
    BLEService *pService = pServer->createService(SERVICE_UUID);

    // Create a BLE Characteristic
    pCharacteristic = pService->createCharacteristic(
        CHARACTERISTIC_UUID,
        BLECharacteristic::PROPERTY_READ   |
        BLECharacteristic::PROPERTY_WRITE |
        BLECharacteristic::PROPERTY_NOTIFY |
        BLECharacteristic::PROPERTY_INDICATE

```

```

        );

//
https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=
org.bluetooth.descriptor.gatt.client\_characteristic\_configuration.xml
// Create a BLE Descriptor
pCharacteristic->addDescriptor(new BLE2902());

// Start the service
pService->start();

// Start advertising
BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
pAdvertising->addServiceUUID(SERVICE_UUID);
pAdvertising->setScanResponse(false);
pAdvertising->setMinPreferred(0x0); // set value to 0x00 to not
advertise this parameter
BLEDevice::startAdvertising();
Serial.println("Waiting a client connection to notify...");
}

void loop() {
    // notify changed value
    if (deviceConnected) {
        pCharacteristic->setValue((uint8_t*)&value, 4);
        pCharacteristic->notify();
        value++;
        delay(10); // bluetooth stack will go into congestion, if too
many packets are sent, in 6 hours test i was able to go as low as 3ms
    }
    // disconnecting
    if (!deviceConnected && oldDeviceConnected) {
        delay(500); // give the bluetooth stack the chance to get
things ready
        pServer->startAdvertising(); // restart advertising
        Serial.println("start advertising");
        oldDeviceConnected = deviceConnected;
    }
}

```

```

}
// connecting
if (deviceConnected && !oldDeviceConnected) {
    // do stuff here on connecting
    oldDeviceConnected = deviceConnected;
}
}

```

Щоб перевірити роботу програми, установимо програму BLE Scanner на смартфон (рис.3.32). Дане програмне забезпечення має наступний вигляд:

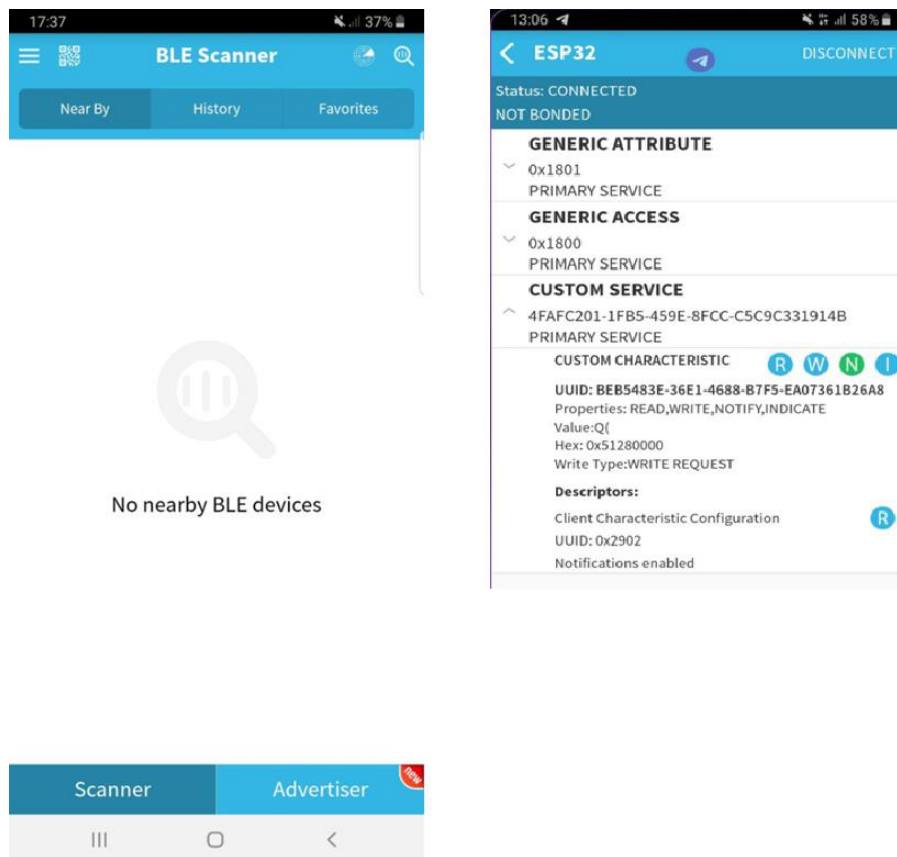


Рисунок 3.32 – Програмне середовище BLE Scanner та підключення до модуля ESP32

Підключаємось до модуля. Для цього серед списку модулів Bluetooth потрібно вибрати той, який задано в коді (ESP32).

Результат підключення, можна також побачити перейшовши в монітор порта в Arduino IDE (рис. 3.34).



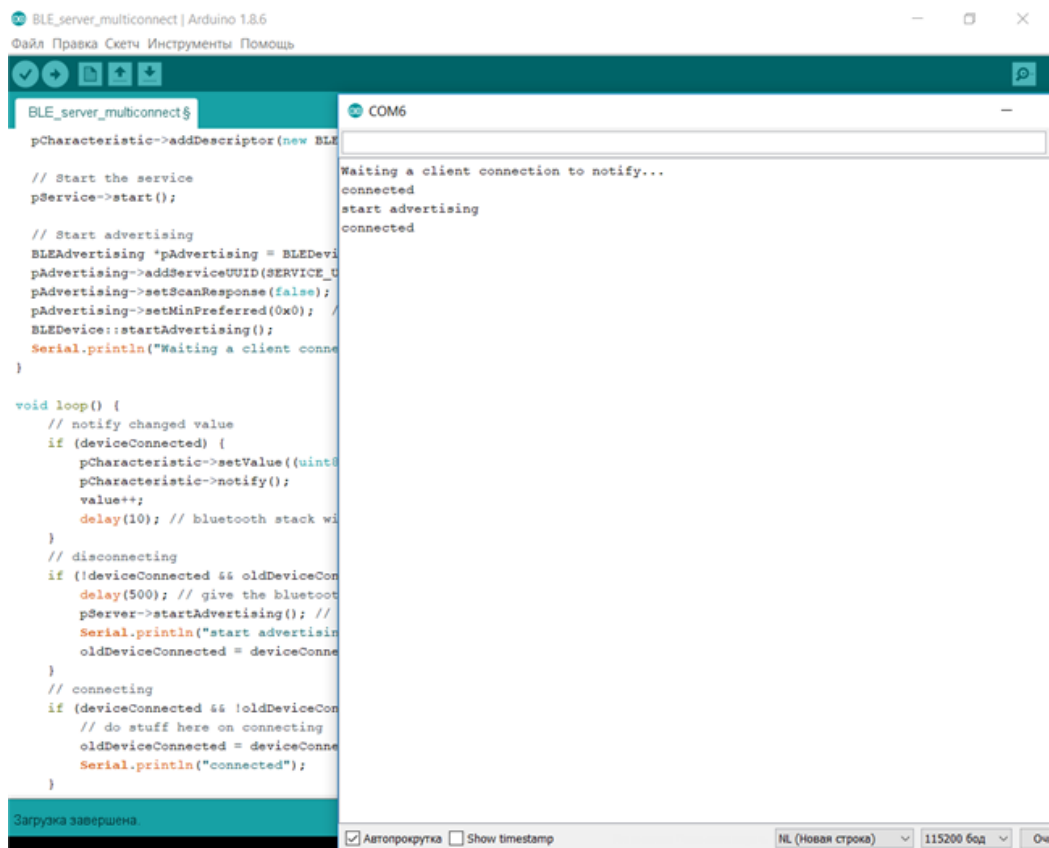


Рисунок 3.34 – Перевірка підключення

### Контрольні питання

1. В чому полягає принцип технології BLE?
2. Назвіть основні блоки Bluetooth-пристроїв.
3. Які основні переваги технології BLE?
4. В яких галузях найчастіше використовують технологію BLE?
5. Які можливості надає додаток BLE Scanner користувачу?

**Підготувати звіт** згідно з ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань).

### **3.7 Лабораторна робота №7. Створення сервера на базі Arduino Uno за допомогою W5100 Ethernet Shield**

**Мета:** дослідити роботу модуля W5100 Ethernet Shield для Arduino.

**Завдання:** прочитати унікальний ID код та температуру датчика DS18S20 та передати їх в комп'ютер через інтерфейс USB.

**Обладнання:** мікроконтролер Arduino UNO R3; проводи папа-папа; модуль Ethernet shield W5100; макетна плата; USB-кабель, плата для прототипування; світлодіоди; резистори (220 Ом).

#### **Загальні відомості**

Arduino Ethernet Shield дозволяє підключити плату Arduino до мережі. Вона заснована на Ethernet-мікросхемі Wiznet W5100. Wiznet W5100 підтримує стеки TCP і UDP в IP-мережі. Підтримує до чотирьох одночасних підключень до сокета. Для створення скетчів, які підключаються до мережі за допомогою даної плати, слід використовувати бібліотеку Ethernet. Дана плата з'єднується з платою Arduino за допомогою довгих стрижнів, що проходять через неї. Це дозволяє не змінювати розташування виводів і встановлювати інші плати поверх неї.

Плата Ethernet Shield має стандартний роз'єм RJ-45 з вбудованим лінійний трансформатором і опцією Power over Ethernet.

6-контактний роз'єм для послідовного програмування сумісний з кабелями і платами-перехідниками FTDI USB. Він підтримує автоматичне скидання, що дозволяє завантажувати скетчі без натискання кнопки скидання на платі. При підключенні через адаптер FTDI-USB, Arduino і Ethernet Shield отримують живлення від адаптера.

Arduino здійснює зв'язок з W5100 і картою SD за допомогою шини SPI (через роз'єм ICSP header). Вона розташована на виводах 11, 12, і 13 плати Uno/Duemilanove і виводах 50, 51, і 52 плати Mega. На обох платах виводи № 10 використовуються для вибору W5100 і введення № 4 – для карти SD. Дані

виводи не можуть бути використані для іншого введення-виведення. На платі Mega, апаратний вивід SS, № 53, не використовується для вибірки ні W5100, ні карти SD, але він повинен бути налаштований на вивід, інакше інтерфейс SPI не працюватиме.

Відзначимо, що оскільки W5100 і карта SD поділяють шину SPI, то одночасно працювати вони не можуть. Якщо ви використовуєте обидва цих периферійних пристрої в своїй програмі, вам слід подбати про відповідні бібліотеки. Якщо ви не використовуєте один з цих периферійних пристроїв, вам слід відключити його. Щоб зробити це, налаштуйте вивід плати 4 як вихід і запишіть в нього "1". Для W5100, встановіть на виводі 10 високий рівень [16].

Ця плата має стандартний роз'єм Ethernet RJ45. Кнопка скидання плати перезапускає і дочірню плату, і плату Arduino.

Плата має кілька індикаторних світлодіодів:

- PWR індикація наявності живлення плати;
- LINK індикація наявності мережевого линка, миготіння при відправці або отриманні даних;
- FULLD індикація повнодуплексного з'єднання;
- 100M індикація з'єднання на швидкості 100 Мб/с (на відміну від з'єднання на 10 Мб/с)
- RX миготить при отриманні платою даних;
- TX миготить при відправці платою даних;
- COLL миготить при мережевий колізії.

### **Хід виконання роботи**

1. Підключити Ethernet Shield до плати Arduino та до мережі.
2. Скласти макет згідно завданню (рис. 3.35).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи макету.

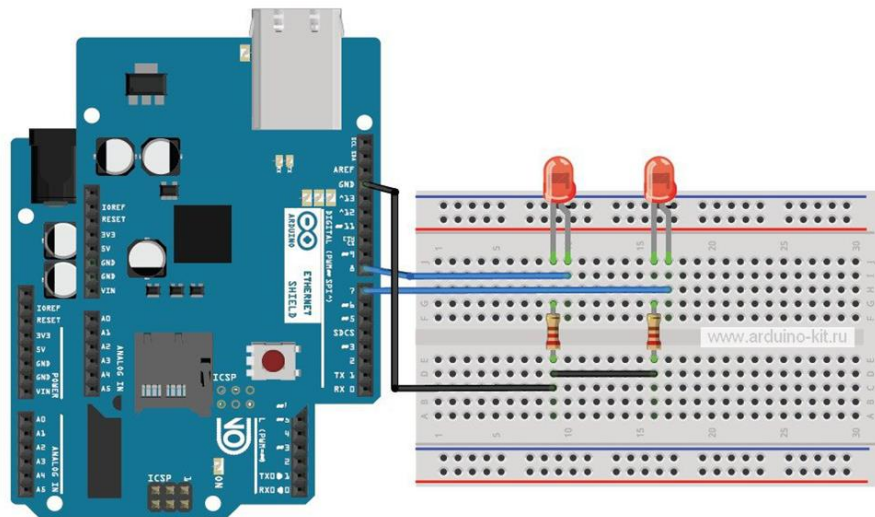


Рисунок 3.35 – Схема макету

### Завдання

Ethernet Shield дозволяє легко підключити плату Arduino до локальної мережі або мережі Інтернет. Він надає можливість Arduino відправляти і приймати дані з будь-якої точки світу за допомогою інтернет-з'єднання. Наприклад, можна реалізувати дистанційне керування вашими виконавчими пристроями, підключеними до реле, через веб-сайт або створити пристрій, який за допомогою звукового сигналу сповістить вас про новий електронний лист.

Підключаємо до плати Arduino Ethernet Shield, а до висновків D7, D8 - світлодіоди через резистор 220 Ом (рис. 3.35).

#### Порядок підключення

1. Підключаємо Ethernet Shield до плати Arduino, за допомогою кабелю RJ45 підключаємо Ethernet Shield до мережі.
2. Підключаємо світлодіоди за схемою на рис. 3.35.
3. Завантажуємо в плату Arduino скетч з лістингу.
4. Відкриваємо браузер на будь-якому комп'ютері даної мережі і в адресному рядку набираємо <http://192.168.0.214> (ту адресу, яку ви привласнюєте Arduino в скетчі).

5. На сторінці (рис. 3.36), змінюючи статус елементів input radio, можемо спостерігати зміну стану світлодіодів, підключених до плати Arduino.

При написанні скетчу використовуємо вбудовану в Arduino IDE бібліотеку Ethernet.

Вміст скетчу показано нижче:

```
#include <SPI.h>

#include <Ethernet2.h>

// mac-адрес плати и ip-адрес сервера
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1,7);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

int pins[] = { 7, 8}; // Пини для світлодіодів
int pinState[] = {0, 0}; // Стан пінів
String getData="";
boolean startGet=false;

void setup()
{
  Serial.begin(9600);
  for(int i=0;i<2;i++)
  {
    pinMode(pins[i],OUTPUT); // контакти підключення світлодіодів
    digitalWrite(i,LOW); // вимкнути світлодіоди
  }
  // ініціалізація бібліотеки Ethernet server
  Ethernet.begin(mac, ip);
  server.begin();
```

```

}
void loop()
{
// очікування підключення клієнтів
EthernetClient client = server.available();
if (client)
{
boolean currentLineIsBlank = true;
while (client.connected())
{
if (client.available())
{
char c = client.read();
if(startGet==true) // дані після '?'
getData+=c;
if(c == '?') // початок збору даних після '?'
startGet=true;
if (c == '\n' && currentLineIsBlank) // закінчення отримання
{
if(getData.length()<1) // запрос без get-данных
{
pinState[0]=0;
pinState[1]=0;
}
else
{
pinState[0]=int(getData[5])-48;
pinState[1]=int(getData[12])-48;
}
}
}
}
}

```

```

// відправка заголовків клієнту
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connection: close");
client.println();
// формування сторінки відповідей
client.println("<!DOCTYPE HTML>");
client.println("<html>");
client.println("<h3>Ethernet shield + LEDS</h3>");
client.println("<form method='get'>");
// світлодіод 1
client.print("<div>");
client.print("led1 off<input type='radio' name='led1' value=0
onclick='document.getElementById(\"submit\").click();' ");
if (pinState[0] == 0)
client.print("checked");
client.println(">");
client.print("<input type='radio' name='led1' value=1
onclick='document.getElementById(\"submit\").click();' ");
if (pinState[0] == 1)
client.print("checked");
client.println("> on");
client.println("</div>");
// світлодіод 2
client.print("<div>");
client.print("led2 off<input type='radio' name='led2' value=0
onclick='document.getElementById(\"submit\").click();' ");
if (pinState[1] == 0)
client.print("checked");
client.println(">");

```

```

client.print("<input      type='radio'      name='led2'      value=1
onclick='document.getElementById(\"submit\").click();' ");

if (pinState[1] == 1)
client.print("checked");
client.println("> on");
client.println("</div>");

client.println("<input      type='submit'      id='submit'
style='visibility:hidden;' value='Refresh'>");

client.println("</form>");
client.println("</html>");

break;

}

if (c == '\n')
{currentLineIsBlank = true;}
else if (c != '\r')
{currentLineIsBlank = false;}
}
}
}

// затримка для отримання клієнтом даних
delay(1);

// закрити з'єднання
client.stop();

for(int i=0;i<2;i++) // світлодіоди ввімкнути або вимкнути
{digitalWrite(pins[i],pinState[i]);}

startGet=false;

getData="";

}

```



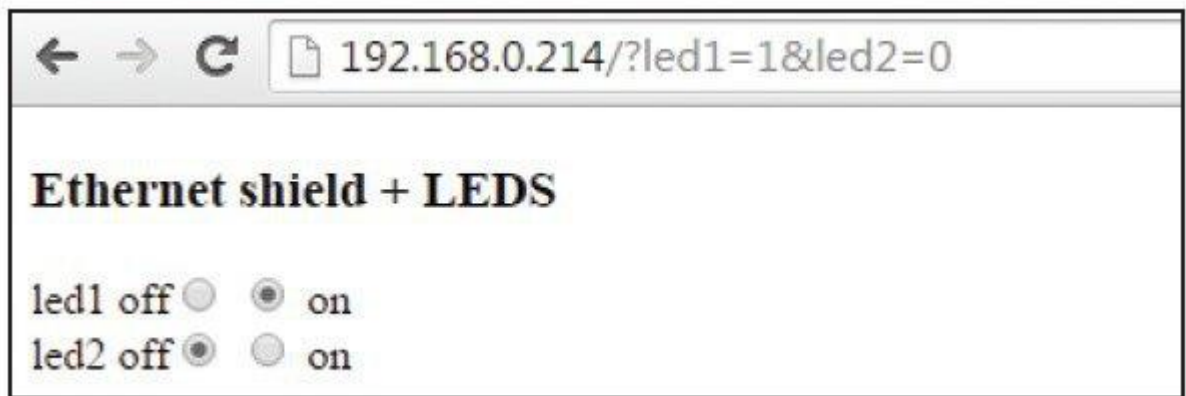


Рисунок 3.36 – Веб-сторінка сформована Arduino-сервером

### Контрольні питання

1. Поясніть призначення модуля W5100 Ethernet Shield?
2. Як підключити модуль Ethernet Shield до плати Arduino?
3. Чи можуть одночасно працювати модуль W5100 і карта SD?
4. Яку бібліотеку слід використовувати для модуля Ethernet Shield?
5. Наведіть основні технічні характеристики модуля Ethernet Shield?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

### 3.8 Лабораторна робота №8. Вивчення роботи модуля LoRa

**Мета:** дослідити роботу модуля LoRa використовуючи контролер Arduino.

**Завдання:** зібрати схему і написати програми що будуть від одного контролера передавати інформацію на інший за допомогою модуля LoRa, детальніше в пункті "Завдання".

**Обладнання:** контролер LoRa на базі модему SX1276; Wi-Fi мікроконтролер ESP32 з OLED дисплеєм; макетна плата; USB-кабель.

#### Загальні відомості

Технологія LoRa (від англ. Long Range) з'явилася на світ під егідою некомерційної організації "LoRa Alliance", заснованої такими компаніями, як IBM, Semtech, Cisco, Kerlink, IMST, MultiTech, Microchip Technology і ін., з метою прийняття та просування протоколу LoRaWAN як єдиний стандарт для глобальних мереж з низьким енергоспоживанням (LPWAN – від англ. Low Power Wide Area Network).

Технологія LoRa об'єднує в собі метод модуляції LoRa в безпроводових мережах LPWAN, розроблений і запатентований Semtech Corporation, і відкритий протокол LoRaWAN, розроблений і представлений в 2015 р. Semtech Corporation і дослідницьким центром IBM Research.

Ключовими перевагами технології LoRa, позиціонуються «LoRa Alliance», є:

- можливість автономної роботи кінцевих пристроїв аж до 10 років від одного акумулятора типорозміру AA за рахунок наднизького енергоспоживання LoRa-модемів (в режимі прийому даних – від 9,7 мА, в режимі передавання – від 40мА, в режимі сну – 200 нА);
- висока стійкість за рахунок можливості демодуляції сигналів на рівні ~ 20dB нижче рівня шумів.

У роботі використано контролер LoRa на базі модему SX1276 і Wi-Fi мікроконтролер ESP32 з OLED дисплеєм. Робоча частота модему LoRa 868-915 МГц при високій чутливості більш -148dBm і вихідної потужності передавача + 20 dBm, що забезпечує високу надійність передавання даних і велику відстань стійкого зв'язку. На борту контролера встановлено 4 Мб пам'яті програм, Wi-Fi антена, 0,96 дюймовий синій OLED дисплей, схема підключення літієвої батареї, схема зарядки, інтерфейс резервної батареї і конвертер USB-UART CP2102 для програмування і передавання даних між комп'ютером і ESP32. Модуль призначено розробки додатків в середовищі розробки ArduinoIDE. Робоча напруга модуля від 3,3 В до 7 В. Зовнішній вигляд модулю наведено на рис. 3.37.

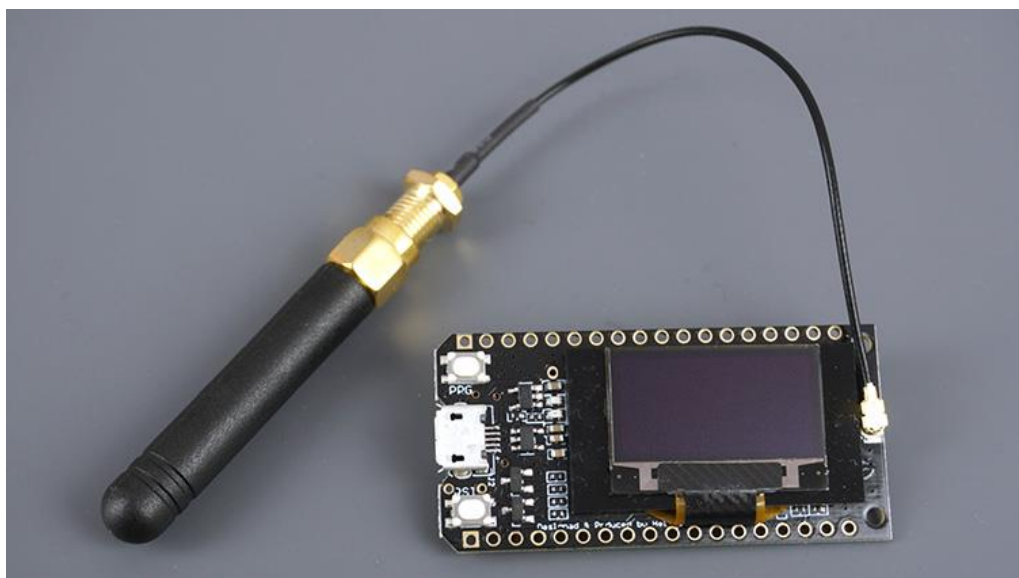


Рисунок 3.37 – Модуль ESP32 з LoRa модемом SX1276 та антеною

В роботі використовується два модулі LoRa.

**Увага!** Електроживлення на модуль можна подавати тільки за підключеної антени LoRa модему! Робота без підключеної антени неминуче призведе до пошкодження підсилювача модуля LoRa.

## **Хід виконання роботи**

1. Підключити модуль ESP32 з LoRa модемом SX1276 та антеною.
2. Скласти макет згідно завданню.
3. Встановити бібліотеки Arduino IDE для здійснення безпроводового зв'язку LoRa.
3. Завантажити програму в мікроконтролер.
4. Перевірити правильність роботи макету.

## **Завдання**

### **1. Встановлення бібліотек Arduino IDE для здійснення безпроводового зв'язку LoRa.**

Після того як антена підключена до плати, а плата підключена до USB-порту комп'ютера можна перейти до Arduino IDE. Для роботи з модулем LoRa, з використанням Arduino, використовується бібліотека ESP32 та LoRa від Sandeep Mistry. Також необхідно встановити бібліотеки для керування OLED-дисплеєм за допомогою ESP32. У роботі використано дві бібліотеки Adafruit: бібліотеку "Adafruit\_SSD1306" та бібліотеку "Adafruit\_GFX".

#### **1.1 Встановлення бібліотеки ESP32.**

В Arduino IDE перейти до File> Preferences. Ввести [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json) в поле "Additional Board Manager URLs" як показано на рис. 3.38. Потім натиснути кнопку "OK".

Якщо вже є URL-адреси плати ESP8266, то потрібно розділити URL-адреси комою.

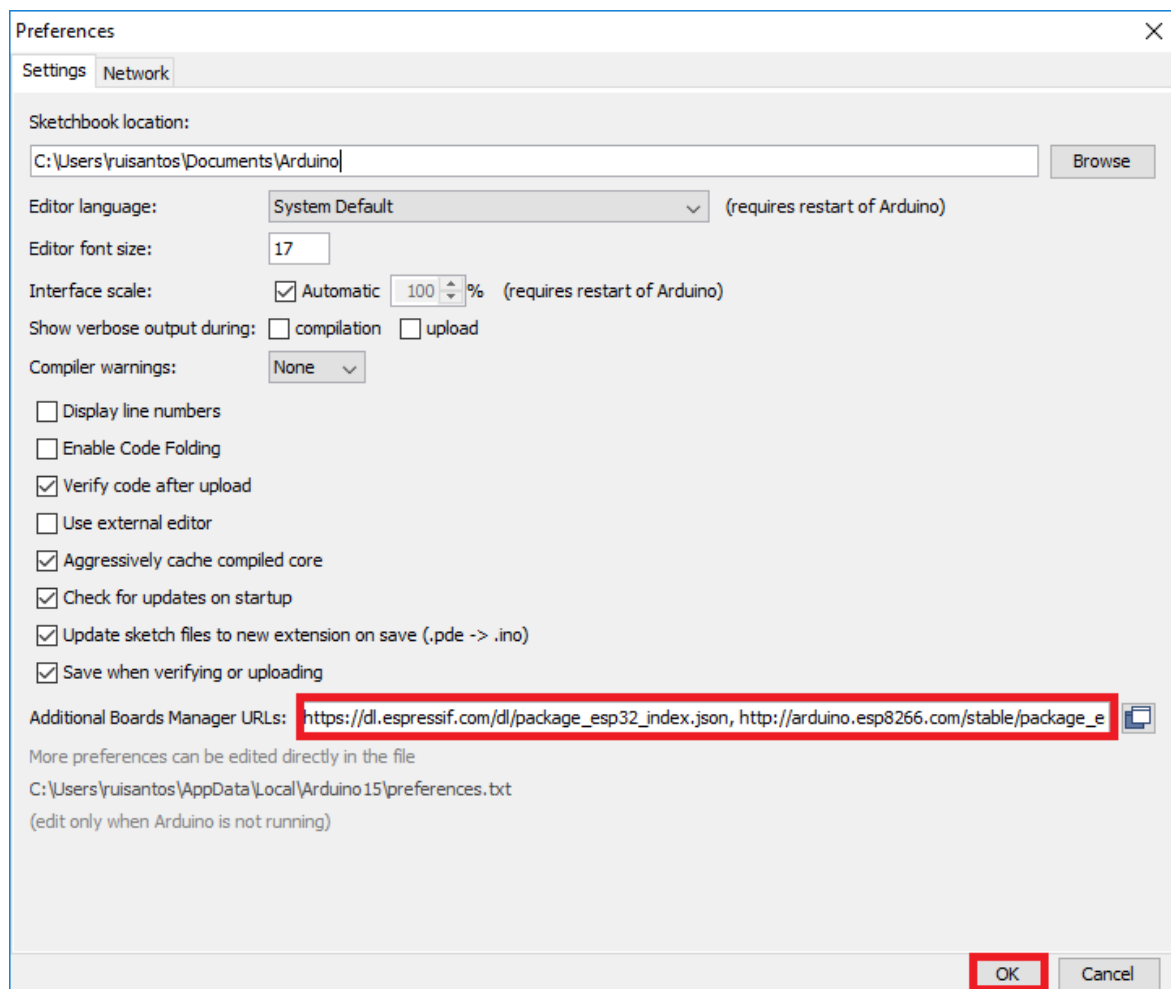


Рисунок 3.38 – Встановлення бібліотеки ESP32

## 1.2 Відкриття Менеджера плат.

Щоб відкрити Менджер плат, потрібно перейти, як показано на (рис. 3.39): Tools > Board > Boards Manager.

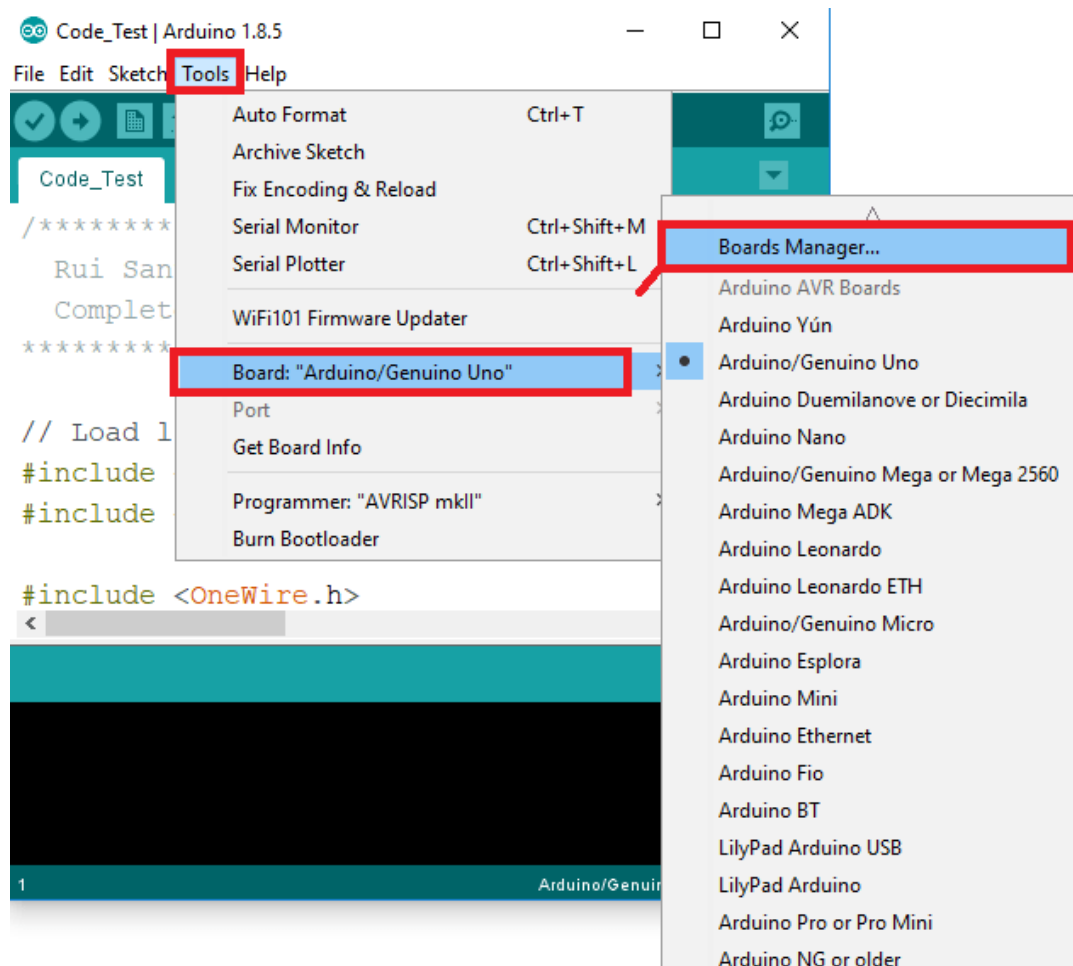


Рисунок 3.39 – Вибір менеджера плат

1.3 Знайти ESP32 і натиснути кнопку встановлення для "ESP32 by Espressif Systems" (рис. 3.40).

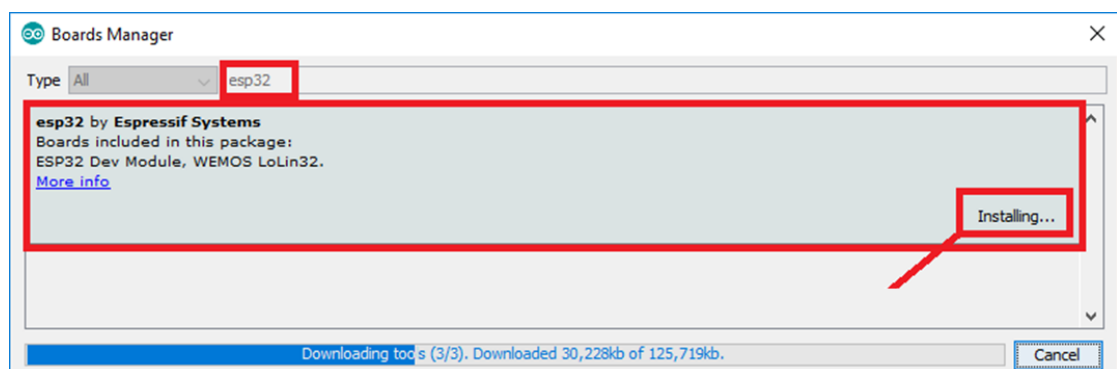


Рисунок 3.40 – Встановлення ПО для плати ESP32

Після інсталяції Менеджер бібліотек буде мати вигляд як на рис. 3.41.

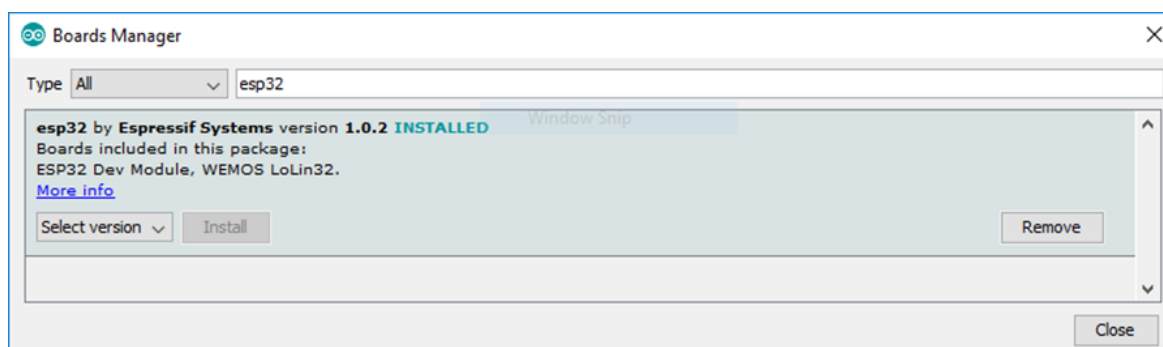


Рисунок 3.41 – Результат встановлення ПО для плати ESP32

Далі встановлюють бібліотеки Adafruit: "Adafruit\_SSD1306", "Adafruit\_GFX" та "LoRa" аналогічно до встановлення ESP32. Відкрити Arduino IDE Sketch > Include Library > Manage Libraries. Введіть "SSD1306" у вікні пошуку та встановіть бібліотеку SSD1306 від Adafruit. Потім таким же чином введіть "SSD1306" у вікні пошуку та встановіть бібліотеку SSD1306. І нарешті встановлюється бібліотека "LoRa". Скріншоти інсталяції у Менеджері бібліотек показані на рис. 3.42 – рис. 3.44.

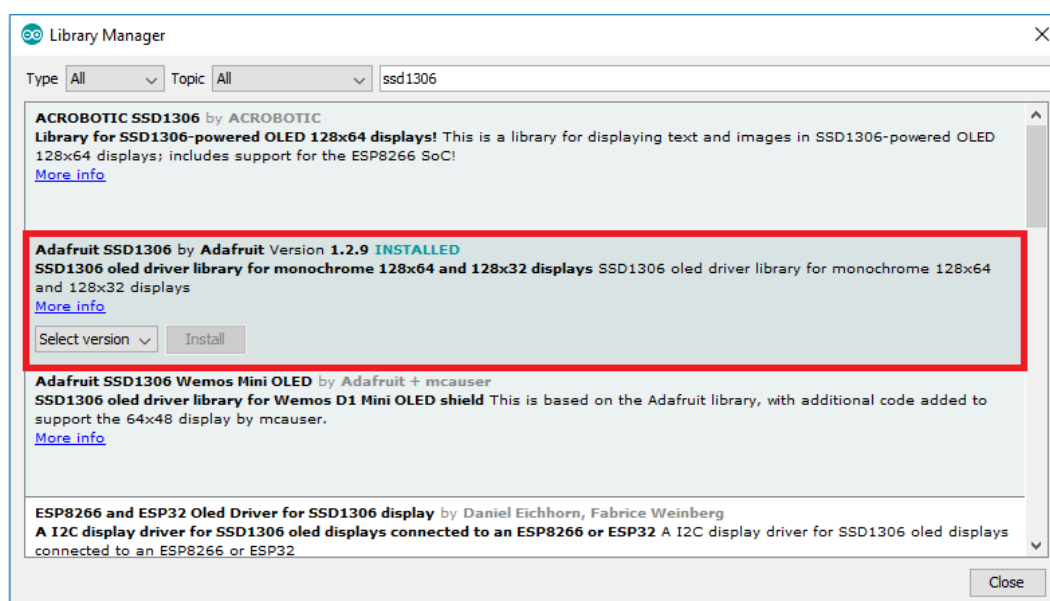


Рисунок 3.42 – Встановлення бібліотеки SSD1306

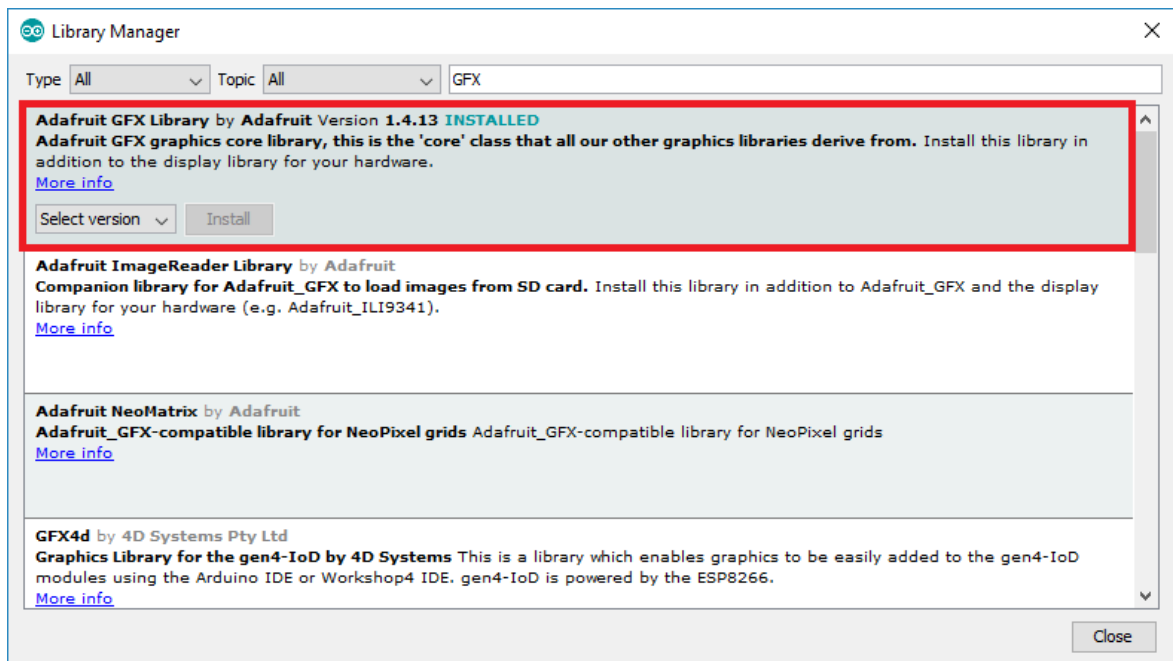


Рисунок 3.43 – Встановлення бібліотеки GFX

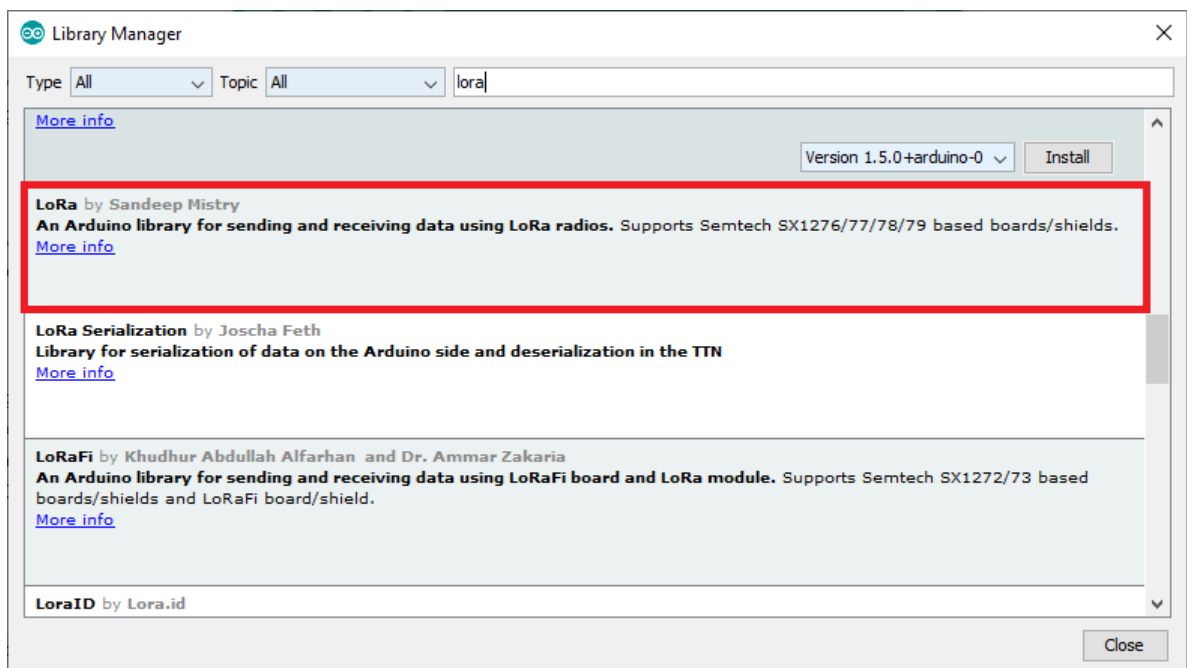


Рисунок 3.44 – Встановлення бібліотеки LoRa

## 2. Приклад використання модуля LoRa

Створимо дві програми Sender та Receiver.



Програма Sender надсилає повідомлення "hello", а потім змінює значення лічильника через протокол LoRa кожні 10 секунд. Він також відображає лічильник на OLED-дисплеї.

```
#include <SPI.h>
#include <LoRa.h>

//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//define the pins used by the LoRa transceiver module
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26

//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 866E6

//OLED pins
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

//packet counter
int counter = 0;
```

```

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RST);

void setup() {

    //reset OLED display via software
    pinMode(OLED_RST, OUTPUT);
    digitalWrite(OLED_RST, LOW);
    delay(20);
    digitalWrite(OLED_RST, HIGH);

    //initialize OLED
    Wire.begin(OLED_SDA, OLED_SCL);
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false,
false)) { // Address 0x3C for 128x32
        Serial.println(F("SSD1306 allocation failed"));
        for(;;); // Don't proceed, loop forever
    }

    display.clearDisplay();
    display.setTextColor(WHITE);
    display.setTextSize(1);
    display.setCursor(0,0);
    display.print("LORA SENDER ");
    display.display();

    //initialize Serial Monitor
    Serial.begin(115200);

    Serial.println("LoRa Sender Test");

    //SPI LoRa pins
    SPI.begin(SCK, MISO, MOSI, SS);

```

```

//setup LoRa transceiver module
LoRa.setPins(SS, RST, DIO0);

if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
}
Serial.println("LoRa Initializing OK!");
display.setCursor(0,10);
display.print("LoRa Initializing OK!");
display.display();
delay(2000);
}

void loop() {
    Serial.print("Sending packet: ");
    Serial.println(counter);

    //Send LoRa packet to receiver
    LoRa.beginPacket();
    LoRa.print("hello ");
    LoRa.print(counter);
    LoRa.endPacket();

    display.clearDisplay();
    display.setCursor(0,0);
    display.println("LORA SENDER");
    display.setCursor(0,20);
    display.setTextSize(1);
    display.print("LoRa packet sent.");
    display.setCursor(0,30);
    display.print("Counter:");
    display.setCursor(50,30);
    display.print(counter);

```

```

display.display();
counter++;
delay(10000);
}

```

## 2.1 Перевірка програми Sender

Щоб завантажити код в плату потрібно вибрати правильну плату та COM-порт.

Щоб вибрати плату, у Arduino IDE необхідно перейти до Tools > Board та вибрати плату TTGO LoRa32-OLED V1, як показано на рис. 3.45.

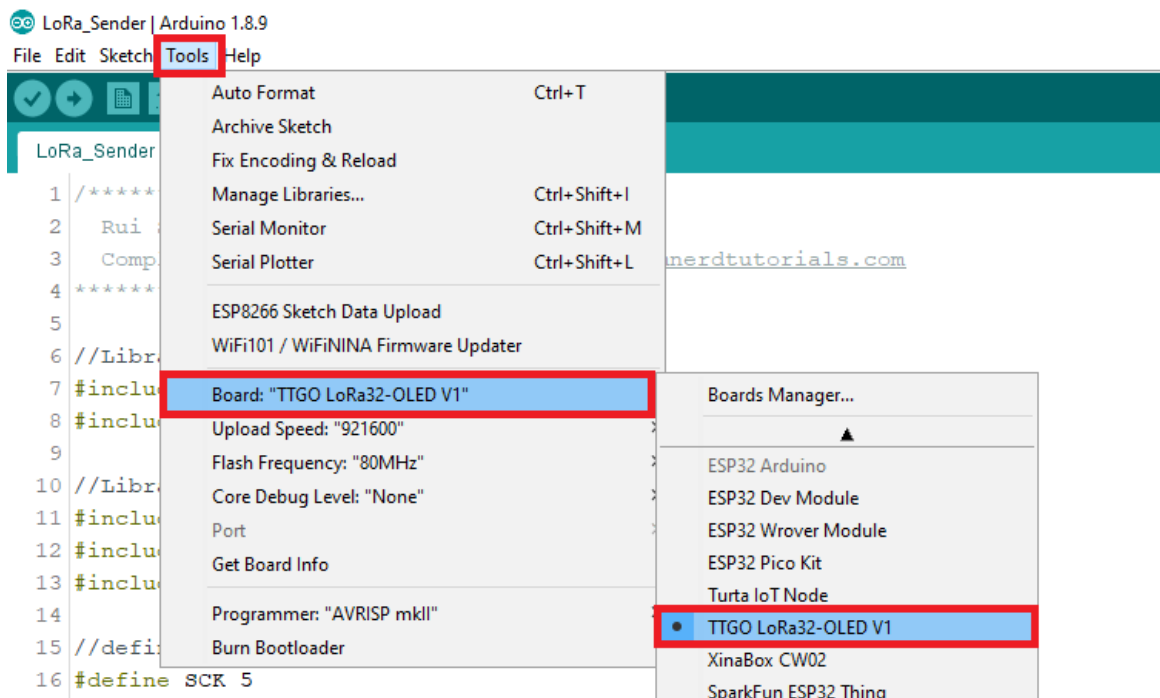


Рисунок 3.45 – Вибір плати TTGO LoRa32-OLED V1

Після завантаження коду в плату, вона почне надсилати пакети LoRa (рис. 3.46).

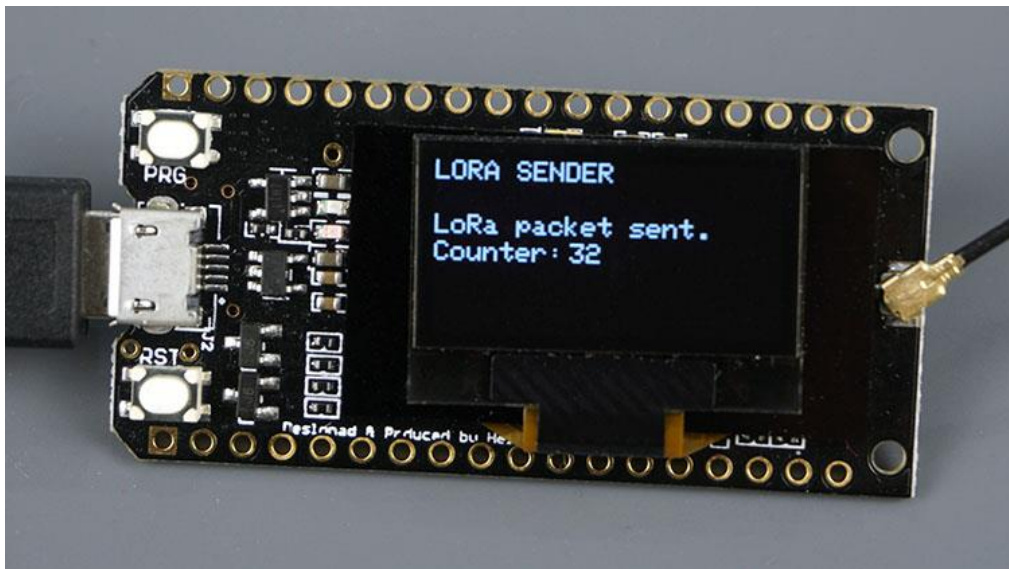


Рисунок 3.46 – Зовнішній вигляд плати Sender у робочому режимі

Тепер необхідно завантажити ескіз програми Receiver на іншу плату OLED TTGO LoRa32 OLED. Ця програма прослуховує пакети LoRa в межах свого діапазону і виводить вміст пакетів на OLED, а також RSSI (відносна потужність прийнятого сигналу).

```
//Libraries for LoRa
#include <SPI.h>
#include <LoRa.h>

//Libraries for OLED Display
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

//define the pins used by the LoRa transceiver module
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DI00 26
```

```

//433E6 for Asia
//866E6 for Europe
//915E6 for North America
#define BAND 866E6

//OLED pins
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RST);

String LoRaData;

void setup() {

    //reset OLED display via software
    pinMode(OLED_RST, OUTPUT);
    digitalWrite(OLED_RST, LOW);
    delay(20);
    digitalWrite(OLED_RST, HIGH);

    //initialize OLED
    Wire.begin(OLED_SDA, OLED_SCL);
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false,
false)) { // Address 0x3C for 128x32
        Serial.println(F("SSD1306 allocation failed"));
        for(;;); // Don't proceed, loop forever
    }
}

```

```

display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(0,0);
display.print("LORA RECEIVER ");
display.display();

//initialize Serial Monitor
Serial.begin(115200);

Serial.println("LoRa Receiver Test");

//SPI LoRa pins
SPI.begin(SCK, MISO, MOSI, SS);
//setup LoRa transceiver module
LoRa.setPins(SS, RST, DIO0);

if (!LoRa.begin(BAND)) {
    Serial.println("Starting LoRa failed!");
    while (1);
}
Serial.println("LoRa Initializing OK!");
display.setCursor(0,10);
display.println("LoRa Initializing OK!");
display.display();
}

void loop() {

    //try to parse packet
    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        //received a packet
        Serial.print("Received packet ");
    }
}

```

```

//read packet
while (LoRa.available()) {
    LoRaData = LoRa.readString();
    Serial.print(LoRaData);
}

//print RSSI of packet
int rssi = LoRa.packetRssi();
Serial.print(" with RSSI ");
Serial.println(rssi);

// Display information
display.clearDisplay();
display.setCursor(0,0);
display.print("LORA RECEIVER");
display.setCursor(0,20);
display.print("Received packet:");
display.setCursor(0,30);
display.print(LoRaData);
display.setCursor(0,40);
display.print("RSSI:");
display.setCursor(30,40);
display.print(rssi);
display.display();
}
}

```

### 3. Тестування LoRa-приймача

Після завантаження коду у плату TTGO LoRa32-OLED V1 він повинен почати отримувати пакети LoRa з іншої плати згідно з рис. 3.47.





Рисунок 3.47 – Зовнішній вигляд плати Receiver у робочому режимі

### Контрольні питання

1. Що таке LoRa?
2. Які головні переваги технології LoRa?
3. Чи можна подавати електроживлення на модуль за відключеної антени LoRa модему?
4. Які бібліотеки необхідно встановити для реалізації безпроводового зв'язку з модулем LoRa?
5. Поясніть алгоритм програм Sender та Receiver?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

### **3.9 Лабораторна робота №9. Підключення модуля ESP8266 NodeMcu V3 до IoT платформи ThingSpeak з використанням AT команд**

**Мета:** підключити модуль ESP8266 NodeMcu V3 до плати Arduino Uno та використовувати AT команди для обміну даними з IoT платформою ThingSpeak..

**Завдання:** зібрати схему і написати програму що буде обмінюватися інформацією з обраним сервером..

**Обладнання:** мікроконтролер Arduino UNO R3; модуль ESP8266 NodeMcu V3; макетна плата; USB-кабель.

#### **Загальні відомості**

NodeMcu – це платформа на основі ESP8266 для створення різних пристроїв Інтернету речей (IoT). Модуль вміє відправляти і отримувати інформацію в локальну мережу або в Інтернет за допомогою Wi-Fi.

Модуль NodeMcu має наступні характеристики:

- підтримує Wi-Fi протокол 802.11 b/g/n;
- підтримувані режими Wi-Fi – точка доступу, клієнт;
- вхідна напруга 3,7 В – 20 В;
- робоча напруга 3В – 3,6;
- максимальний струм 220 мА;
- вбудований стек ТС /IP;
- діапазон робочих температур від -40 °С до 125 °С;
- 80 МГц, 32-бітний процесор;
- час пробудження і відправки пакетів 22 мс;
- наявність регуляторів, систем управління живленням [17].

## **Хід виконання роботи**

1. Підключити модуль ESP8266 NodeMcu V3 до плати Arduino Uno та до IoT платформи ThingSpeak;
2. Скласти макет згідно завданню (рис. 3.48).
3. Завантажити програму в мікроконтролер Arduino.
4. Перевірити правильність роботи макету.

## **Завдання**

### **1. Підключення ESP8266 NodeMcu V3 до Arduino Uno**

Як відомо, Arduino Uno, Mega або Nano не має мережевих можливостей. Щоб Arduino підключився до Інтернету, йому потрібен модем. Модуль ESP8266 буде виконувати роль такого модему, встановлюючи з'єднання з локальним Wi-Fi-роутером для того, щоб надсилати або отримувати дані з Інтернету.

Для керуванням модулем ESP8266 використовують AT команди. Команди AT – це інструкції, що використовуються для управління модемом. AT – це аббревіатура Attention. Кожен командний рядок починається з "AT" або "at". Тому команди модему називаються командами AT.

Необхідно зауважити, що початковий "AT" – це префікс, який інформує модем про початок командного рядка. AT команди або Hayes використовуються не тільки ESP8266, але й іншими модемами, такими як GSM, Bluetooth та GPRS. Для того, щоб плата Arduino Uno могла надсилати команди модулю ESP8266 та приймати відповіді від нього, їх потрібно з'єднати згідно схемі на рис. 3.48

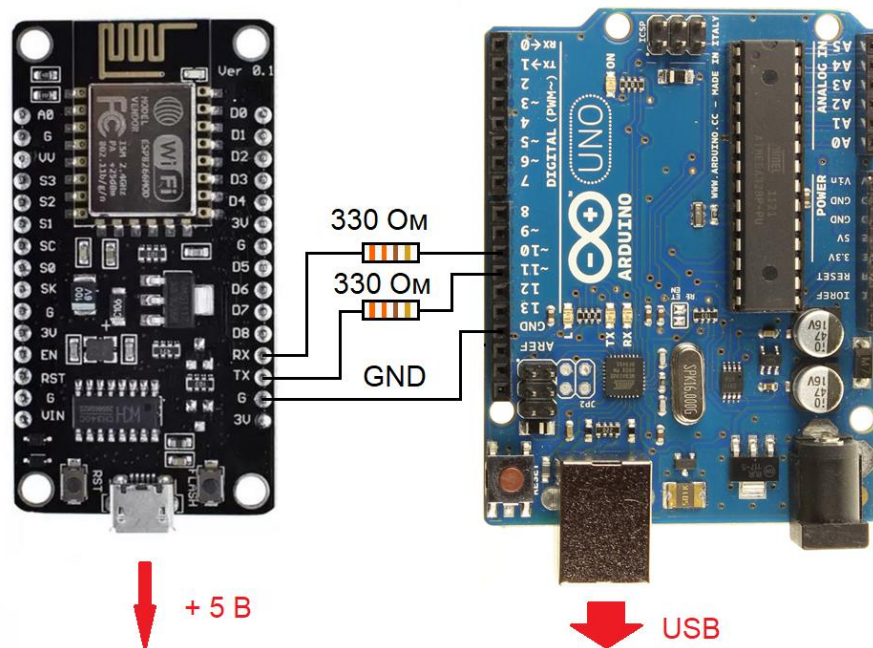


Рисунок 3.48 – Під'єднання модулю ESP8266 до плати Arduino Uno

Модулі ESP8266, зокрема модуль NodeMcu V3, працюють від живлення 3,3 В. Таким чином, не можна підключати вихідні виводи плати Arduino Uno до виводів ESP8266. На малюнку вказані резистори номіналом 330 Ом, які використані щоб зменшити логічний рівень напруги в 5 В. Також необхідно зауважити, що для правильної роботи модуля ESP8266 NodeMcu V3 він повинен бути підключений до окремого джерела живлення, у якості якого виступає USB-порт комп'ютера.

## 2. Підключення до IoT платформи ThingSpeak

Спершу необхідно створити кінцеву точку, до якої можна надіслати деякі дані. Для цього потрібно зробити наступне:

- створити обліковий запис на ThingSpeak <https://thingspeak.com/> (рис.3.49);
- створити канал з одним полем label;
- отримати ключ API;
- переглянути URL-адресу " Update a Channel Feed ";



Рисунок 3.49 –Головна сторінка IoT платформи ThingSpeak

Коли канал створено, його сторінка має наступний вигляд (рис. 3.50).

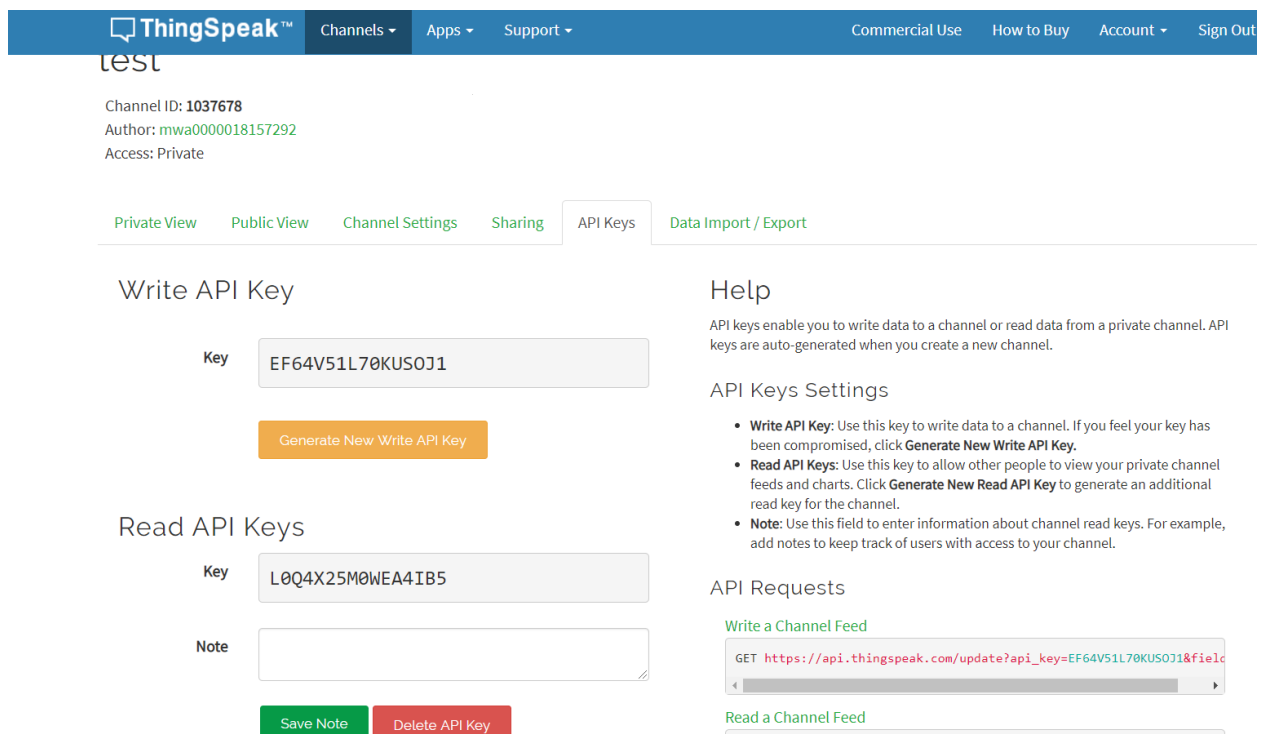


Рисунок 3.50 – Сторінка налаштувань каналу платформи ThingSpeak

Використовуючи отриманий ключ, можна зробити тестовий HTTP запит до сервера ThingSpeak:

[https://api.thingspeak.com/update?api\\_key=YOUR\\_KEY\\_HERE&field1=4](https://api.thingspeak.com/update?api_key=YOUR_KEY_HERE&field1=4)

Цей запит надіслає значення 4 у поле1. Необхідно це зробити у браузері, а потім переглянути дані в приватному перегляді створеного каналу.

### **3. AT команди**

Після того, як модуль ESP8266 NodeMcu V3 підключено до Arduino, AT команди можуть бути надіслані через послідовний порт. Для досягнення мети, яка полягає в підключенні до Інтернету, потрібен лише певний набір команд AT. Більш ретельне пояснення кожної команди можна знайти нижче.

AT + CIPMUX = 1 Увімкнути єдине (0) або багаторазове підключення (1) до веб-сервера. Багаторазове з'єднання - хороший варіант, якщо потрібно неодноразово надсилати чи читати дані з Інтернету.

AT + CWMODE = 1 Встановити режим Wi-Fi: 1 – станційний режим (ESP8266 - клієнт), 2 - режим AP (ESP8266 діє як маршрутизатор Wi-Fi, до якого можна підключити телефон або ПК), 3 – режим AP + станція (змішаний режим роботи ESP8266).

AT + CWJAP = "< Wi-Fi -імя>,< Wi-Fi - пароль >" Підключіться до свого Wi-Fi. Введіть своє ім'я та пароль SSID всередині подвійних знаків.

AT + CIFSRR - повертає IP-адресу модуля, вказуючи на те, що він успішно підключений до вибраного Wi-Fi роутера.

AT + CIPSTART = 0, "TCP", "www.teachmemicro.com", "80"

Запустити TCP або UDP – з'єднання. Тут перший параметр (0) – це ідентифікатор підключення, "TCP" означає, що ми використовується TCP замість UDP, потім адресу (або ip) веб-сервера, а потім номер порту.

AT + CIPSEND = 0,16 Команда, яка повідомляє, що дані модуля, готові до відправлення. '0' тут – ідентифікатор з'єднання, а '16' – довжина даних, що надсилаються. Після цієї команди ESP8266 відповість символом ">", щоб

сказати, що буде чекати надсилання даних. У разі успіху модуль відповість "Надіслати ОК".

Необхідно звернути увагу, що ці команди повинні бути відправлені в правильному порядку з Arduino в модуль ESP8266.

#### **4. Приклад ескізу з використанням AT команд**

Використовуючи команди AT, описані вище, можна зробити простий HTTP-запит, який дасть можливість передати дані на сервер ThingSpeak.

Нижче наведено ескіз програми:

```
#include <SoftwareSerial.h>
#define RX 11
#define TX 10
String AP = "WIFI_NAME";          // CHANGE ME
String PASS = "WIFI_PASSWORD";    // CHANGE ME
String API = "YOUR_API_KEY";      // CHANGE ME
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;
SoftwareSerial esp8266(RX,TX);

void setup() {
    Serial.begin(9600);
    esp8266.begin(9600);
    sendCommand("AT", 5, "OK");
    sendCommand("AT+CWMODE=1", 5, "OK");
    sendCommand("AT+CWJAP=\"" + AP + "\", \"" + PASS
+"\"", 20, "OK");
}

void loop() {
    //sendCommand("AT", 5, "OK");
```

```

        //delay(2000);
        valSensor = getSensorData();
        String getData = "GET /update?api_key="+ API +"&"+ field
+""+String(valSensor);
        sendCommand("AT+CIPMUX=1",5,"OK");
        sendCommand("AT+CIPSTART=0,\"TCP\", \"\"+ HOST +"\", "+
PORT,15,"OK");
        sendCommand("AT+CIPSEND=0,"
+String(getData.length()+4),4,">");
        esp8266.println(getData);delay(1500);countTrueCommand++;
        sendCommand("AT+CIPCLOSE=0",5,"OK");
    }

    int getSensorData(){
        return random(1000); // Replace with
    }

    void sendCommand(String command, int maxTime, char
readReplay[]) {
        Serial.print(countTrueCommand);
        Serial.print(". at command => ");
        Serial.print(command);
        Serial.print(" ");
        while(countTimeCommand < (maxTime*1))
        {
            esp8266.println(command);//at+cipsend
            if(esp8266.find(readReplay))//ok
            {
                found = true;
                break;
            }

            countTimeCommand++;
        }
    }

```



```

if(found == true)
{
    Serial.println("OYI");
    countTrueCommand++;
    countTimeCommand = 0;
}

if(found == false)
{
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
}

found = false;
}

```

Після того, як скетч завантажено в плату Arduino Uno можна запустити Монітор послідовного порту, щоб перевірити роботу програми (рис. 3.51).

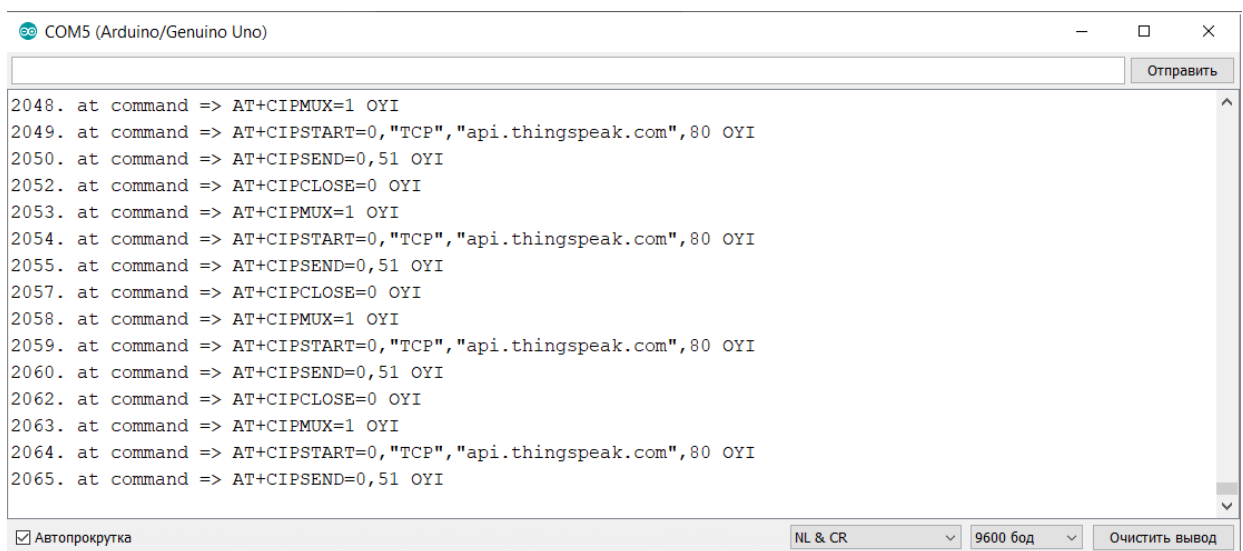


Рисунок 3.51 – Вікно монітору послідовного порту

Тепер можна перейти до сторінки ThingSpeak, щоб побачити прийняті дані рис. (3.52).

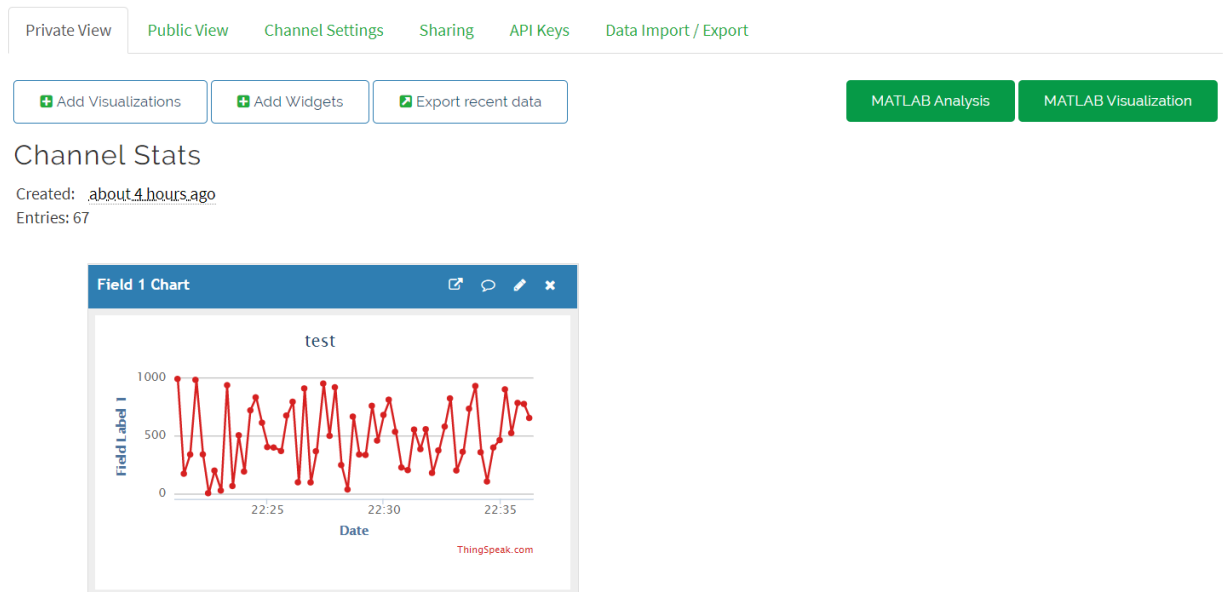


Рисунок 3.52 – Вікно відображення прийнятих даних платформи ThingSpeak

### Контрольні питання

1. Поясніть призначення платформи ThingSpeak?
2. Назвіть основні характеристики ESP8266 NodeMcu V3?
3. Які команди використовують для керування модулем ESP8266?
4. Які АТ команди необхідно використати для підключення модуля ESP8266 NodeMcu V3 до Інтернету?
5. Поясніть алгоритм підключення модуля до IoT платформи ThingSpeak?

**Підготувати звіт** згідно ДСТУ 3008-95 (лістинг програми, висновки, перелік посилань)

## ПЕРЕЛІК ПОСИЛАНЬ

1. Інтернет речей (Internet of Things, IoT). URL: <https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot>. (Дата звернення: 03.03.2020).
2. Buyya R.. Internet of Things Principles and Paradigms. 2016. pp. 9. URL: <http://www.buyya.com/papers/IoT-Book2016-C1.pdf>. (Дата звернення: 04.03.2020).
3. Overview of AWS – Amazon Web Services. 2014. URL: [https://media.amazonwebservices.com/AWS\\_Overview.pdf](https://media.amazonwebservices.com/AWS_Overview.pdf). (Дата звернення: 04.03.2020).
4. IoT platforms: enabling the Internet of Things. 2016. URL: <https://cdn.ihs.com/www/pdf/enabling-IOT.pdf>. (Дата звернення: 05.03.2020).
5. Networking Protocols and Standards for Internet of Things. URL: [https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot\\_prot.pdf](https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot.pdf). (Дата звернення: 05.03.2020).
6. Гойхман В., Лаврова А.. Протокол MQTT. Особенности, варианты применения, основные процедуры MQTT Protocol.. URL: <http://lib.tssonline.ru/articles2/fix-corp/protokol-mqtt-osobennosti-varianty-primeneniya-osnovnye-protsedury-mqtt-protocol>. (Дата звернення: 11.03.2020).
7. Bluetooth Low Energy (BLE) – Cypress Semiconductor. URL: <https://www.cypress.com/file/220246/download>. (Дата звернення: 12.03.2020).
8. Алексеев В.. Новые модули Bluetooth 4.0 серии BLE производства Bluegiga. URL: <https://wireless-e.ru/radiomoduli/ble112/>. (Дата звернення: 12.03.2020).
9. LoRaWAN – LoRa Alliance. URL: <https://loralliance.org/sites/default/files/2018-04/what-is-lorawan.pdf>. (Дата звернення: 13.03.2020).
10. Ethernet и промышленные сети. URL: <http://www.picad.com.ua/0404/pdf/12.pdf>. (Дата звернення: 13.03.2020).
11. Датчики давления Arduino bmp280, bmp180, bme280. URL: <https://arduinomaster.ru/datchiki-arduino/datchiki-atmosfernogo-davleniya-bmp280-bmp180-bme280/>. (Дата звернення: 13.03.2020).
12. Node RED Programming Guide – Programming the IoT. URL: <http://noderedguide.com/>. (Дата звернення: 15.03.2020).

13. Thingier.io – Open Source IoT Platform. URL: <https://thingier.io/>. (Дата звернення: 20.03.2020).
14. ESP8266 Technical Reference - Espressif Systems. URL: [https://www.espressif.com/sites/default/files/documentation/esp8266-technical\\_reference\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf). (Дата звернення: 24.03.2020).
15. Ubidots: IoT platform | Internet of Things. URL: <https://ubidots.com/>. (Дата звернення: 24.03.2020).
16. Arduino Ethernet Shield. URL: [https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100\\_Datasheet\\_v1\\_1\\_6.pdf](https://www.sparkfun.com/datasheets/DevTools/Arduino/W5100_Datasheet_v1_1_6.pdf). (Дата звернення: 30.03.2020).
17. Начало работы с ESP8266 NodeMcu v3 Lua с WiFi. URL: <https://arduino-master.ru/platy-arduino/esp8266-nodemcu-v3-lua/>. (Дата звернення: 30.03.2020).